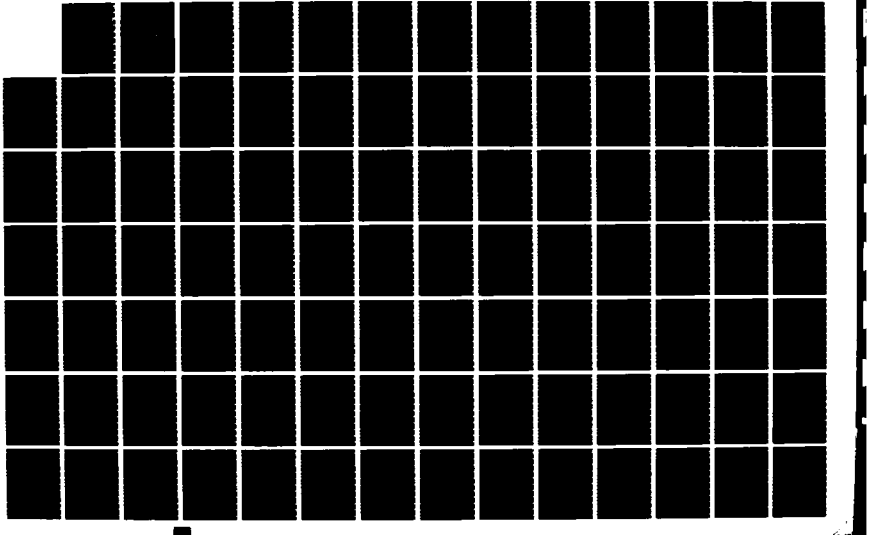
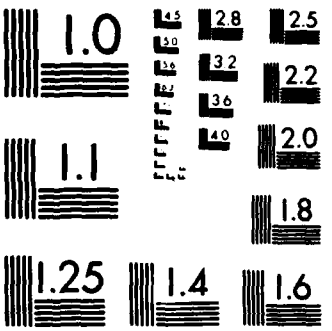
AD-A172 315 CONVERSION OF THE CALAP (COMPUTER AIDED LANDFORM ANALYSIS PROGRAM) PROGRAM FROM FORTRAN TO DUCK(U) JAYCOR ALEXANDRIA VA SEP 86 ETL-0419 DAAK70-82-C-0027 1/2

UNCLASSIFIED F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
32
36
40

AD-A172 315

ETL-0419

CONVERSION OF THE CALAP PROGRAM FROM FORTRAN TO DUCK

FINAL REPORT

ETL-0419

JAYCOR

DTIC FILE COPY

DTIC ELECTE SEP 23 1986 S A

Approved for public release.
Distribution is unlimited.

86 9 23 026

205 South Whiting Street
Alexandria, Virginia 22304

(2)

CONVERSION OF THE CALAP PROGRAM FROM FORTRAN TO DUCK

FINAL REPORT

ETL-0419

Prepared for:

U.S. Army Engineer Topographic Laboratory
Fort Belvoir, Virginia

Under MERADCOM Contract DAAK70-82-C-0027

Prepared by:

Smart Systems Technology, Inc.
Suite 300, 6870 Elm Street
McLean, Virginia 22101

Under Subcontract to:

JAYCOR
205 South Whiting Street
Alexandria, Virginia 22304

DTIC ELECTE SEP 2 3 1986 A

September 1983

This document has been approved for public release and sale; its distribution is unlimited.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-A172 315

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ETL-0419	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): CONVERSION OF THE CALAP PROGRAM FROM FORTRAN TO DUCK	5. TYPE OF REPORT & PERIOD COVERED: Final Report August 1982 - August 1983
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s): DAAK70-82-C0027
9. PERFORMING ORGANIZATION NAME AND ADDRESS: JAYCOR, 205 South Whiting Street, Alexandria, Virginia 22304	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia 22060-5546	12. REPORT DATE: September 1986
	13. NUMBER OF PAGES: 147
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

CALAP
DUCK
artificial intelligence
logic programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An expert advisor program named CALAP (Computer Aided Landform Analysis Program) was developed by Dr. Robert Leighty, working at the U.S. Army Engineer Topographic Laboratories. The original program was developed in FORTRAN on an HP-1000, a minicomputer. CALAP was reprogrammed in an Artificial Intelligence (AI) language called DUCK. DUCK is a Logic Programming Language developed at Yale by Prof. Drew McDermott. CALAP has previously been translated into another AI language called OPS5. The experience of translating CALAP to DUCK is documented here.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Table of Contents

I. Introduction.....1

II. Logic Programming Languages.....5

III. The Structure of CALAP.....11

IV. DUCK Programming.....14

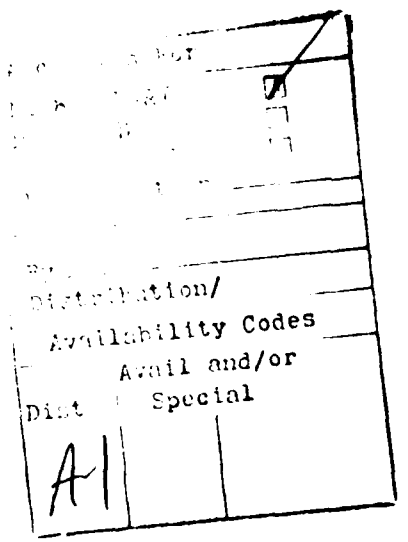
V. Computer Explanation.....19

VI. Conclusion.....22

VII. References.....26

VIII. Appendix.....27

QUALITY INSPECTED 1



ACKNOWLEDGEMENTS

This is the Final Report on the final phase of JAYCOR's activities for the U.S. Army Engineer Topographic Laboratory under MERADCOM Contract Number DAAK70-82-C-0027. SMART SYSTEMS TECHNOLOGY, Inc. performed all system development under subcontract to JAYCOR.

SMART SYSTEMS TECHNOLOGY greatly appreciates the interest and support of Dr. Robert Leighty and Ms. Connie Wickham at the Artificial Intelligence Center, U.S. Army Engineer Topographic Laboratory, Fort Belvoir.

I. Introduction

An expert advisor program named CALAP, (Computer Aided Landform Analysis Program), was developed by Dr. Robert Leighty, working at USAETL. The original program was developed in FORTRAN on an HP-1000, a small minicomputer. CALAP was reprogrammed in an Artificial Intelligence (AI) language called DUCK [1]. DUCK is a Logic Programming Language developed at YALE by Prof. Drew McDermott. CALAP has previously been translated into another AI language [2] called OPS5 [3]. The experience of translating CALAP to DUCK will be documented. This report is similar to [2] and the DUCK version of CALAP was deliberately implemented in a manner as close to OPS5 CALAP as possible so they could be easily compared. In particular, DUCK CALAP was implemented using "forward chaining" (explained in Section II) because OPS5 is a forward production system. The issues of how an intelligent advisor program can or should explain its actions or reasoning will be discussed.

Deductive retrieval and logic programming systems are designed to facilitate the expression of problems requiring inference, problem solving, plan generation, and retrieval of facts from databases, in a formal language. The systems embody algorithms (called resolution, or unification) that enable the computer to treat logical inference [4] as a computation, thereby "mechanizing" the process of formal reasoning.

The mechanization of problem solving began in earnest with the discovery of the resolution algorithm [5] by Alan Robinson in the 1960s. This fusion of mathematical logic and computer science led to the development of many computer programs. For example, Micro-PLANNER and CONNIVER were developed at MIT and QA4 and QLISP were developed at STANFORD in the early 1970s. Similar efforts were underway in Europe.

Recently, the notion of programming in logic [6] has been gaining favor. One advantage of logic programming is tnat it is (or is closer to) a non-procedural representation. This means that logic expresses the "what" (i.e. what needs to happen or be true) of a process without necessarily expressing the "how" (i.e. which order things should be done or proven) of a process. Now a programmer is free to pursue the true structure of a problem without worrying about the procedural details. There are two disadvantages. One, people usually do better with procedural representations, so they must be retrained to be effective logic programmers. Two, since the programmer does not specify the "how" of a process, the logic programming system does. There are many inefficient ways to generate a proof. Many problems have very fast procedural solutions, but very slow non-procedural solutions. All the logic programming systems allow escapes into some procedural language to eliminate such gross inefficiencies. Still, the elegance of the "non-procedural" description has been

marred. See [7] for a discussion of some of the problems with simple logic programming systems.

All logic programming systems implement a (usually in-core) relational database to store the predicates associated with atomic assertions (facts) and theorems (rules). The use and availability of a relational database greatly simplifies the descriptions of and solutions to many problems.

Another advantage of using a logic programming language to construct a system is that a rather natural modularity results. Since the description is non-procedural, any particular piece of knowledge tends to be defined only once. It is the logic programming system's responsibility to access the knowledge when necessary. This modularity means that it is much easier to debug a logic programming system than a system written in a sequential language because a change need only be made once, instead of in all the different places a particular fact could be used.

Another advantage of logic programming is that the definition of a single concept may be equivalent to several function definitions in traditional languages. For example, in LISP, APPEND has two arguments, each a list and returns a single list that is the two lists concatonated together. The equivalent predicate in logic would have three arguments corresponding to the two input arguments and the answer. (Predicates never return an answer; they are true.) In

LISP, (append '(1) '(2)) returns '(1 2). In logic, the query "(append '(1) '(2) ?x)" returns x = '(1 2). But also, the query "(append '(1) ?x '(1 2))" returns x = '(2). The query "(append ?x ?y '(1 2))" would return the answers x = '() and y = '(1 2), x = '(1) and y = '(2), and x = '(1 2) and y = '(). Logic programs tend to be very compact since the predicates can be used in different ways. Unfortunately, certain patterns of variables can result in extremely inefficient computations.

II. Logic Programming Languages

The three most prominent logic programming languages are LOGLISP [8], PROLOG [9,10], and DUCK. Alan Robinson's research has lead to the development of LOGLISP, a system that integrates logic and LISP into one environment. PROLOG implements a computationaly extended logic programming system with a control strategy that is quite efficient in most situations. Drew McDermott's research (extensions of micro-PLANNER and CONNIVER) has lead to the development of DUCK, a system that integrates first order logic, non-monotonic logic, multiple databases, and LISP into one environment.

In LOGLISP, the logic programming system is implemented as a library of LISP routines that can be called by other LISP routines. Likewise, logic programming constructs are able to call LISP routines. The system supports a single global database for assertions and rules. The system only supports backward chaining theorems (backward vs. forward chaining will be discussed under DUCK). The system has automatic proof explanation features.

PROLOG is usually implemented in LISP or PASCAL. Newer PROLOGs allow the use of foreign language routines. In principle, there is one global database and only backward chaining theorems. In practice, one can escape the restrictions, but not efficiently or elegantly.

DUCK, as LOGLISP, is implemented as a library of LISP routines. As in LOGLISP, LISP functions can call logic routines and logic routines (under control of theorems) can call LISP functions. LISP functions can be called during either forward or backward chaining; this will be explained shortly. LISP data structures, symbols, lists, etc., can be represented and manipulated in DUCK. LISP arithmetic functions are directly available in DUCK theorems. DUCK is the most powerful and general environment of the three systems. The control strategies of LOGLISP and PROLOG (and MYCIN, for that matter) can be easily simulated in DUCK, but the features of DUCK cannot be simulated in the other systems without a great deal of effort. The principal features of DUCK will be enumerated.

DUCK supports multiple databases through a facility called "data pools." This allows hypothetical universes to be constructed and deductions to be performed in relative isolation. This is a very useful feature for problem solving.

DUCK supports both forward and backward chaining of theorems. If A is true and A => B is true, then one can deduce B. If B is asserted into the database when both A and A => B are in the database, then we say that B was asserted by forward chaining. Forward chaining causes databases to increase in size. For example, if B => C had been in the database, the C would also have been asserted, etc. If we queried the database about B, the system would

respond that B was true because it would be in the database. With backward chaining, B would not have been asserted into the database. Instead, when we queried the database about B, it would look for, but not find B. It would lookup all the theorems that have B as a consequent and back chain on the antecedants. In our example, the system would find A => B and back chain on A. A is in the database, therefor B is true. Likewise, if B => C is in the database, then a query about C will succeed because C is true if B is, and B is true if A is, and A is true. Backward chaining is a search process that takes time and uses temporary space. Forward chaining takes very little time but uses permanent space. This is the classic space/time tradeoff. DUCK gives the programmer the choice of how theorems are chained. PROLOG and LOGLISP only use backward chaining.

DUCK uses data dependencies to implement a reason maintenance system. The system uses the data dependencies to remember that, for example, B is true because both A and A => B are true. B depends on A and A => B. DUCK uses data dependencies to maintain internal consistency. If either A or A => B is erased (and there is no other independent support for B), then DUCK would erase B (and any consequents that follow from B). This enables a DUCK programmer to do provisional reasoning by asserting and erasing predicates while DUCK takes care of the bookkeeping necessary to maintain consistency.

DUCK supports common-sense reasoning by handling exceptions. It is often the case that a rule is true 99 percent of the time, but that there are exceptions. Cars have four wheels; professors have PhD. degrees; are examples. If one has a 1000 different cars (998 of which have 4 wheels), one does not want to declare 1000 "auto" predicates (e.g. (auto chevy)), 998 "has-4-wheels" predicates, and 2 "not has-4-wheels" predicates. DUCK allows you to declare the 1000 "auto" predicates, the 2 "not has-4-wheels" predicates, and the general rule that (has-4-wheels ?x) is true for ?x if (auto ?x) is true and (not (has-4-wheels ?x)) is not present in the database. The single common-sense rule saves us 998 extra explicit assertions.

DUCK supports non-monotonic reasoning. In pure first order predicate calculus, one can deduce a fact from the explicit existence of another fact or its negation. One cannot deduce a fact from the absence of another fact. Also, any fact that is deduced is expected to be "timelessly" true. You cannot deduce a person's income or weight because they can change depending on what is in the database. Almost any interesting problem requires that such things be computed. DUCK allows a system to react to the presence and absence of predicates in the database through pseudo-predicates with names like THNOT, THCOND, THALL, and CONSISTENT. (thnot ?x), for example, succeeds when ?x fails to unify with anything in the database.

DUCK supports syntactic data-type checking of predicates. DUCK has an abstract data typing facility that helps to keep errors out of the database. For example, a data type VEHICLE can be declared and car, truck, and tank can all be declared to be VEHICLEs. If we declare the predicate "owns" to take one argument that is a VEHICLE, then DUCK will happily accept the assertion (owns truck). DUCK will complain about (owns trunk), (owns motorcycle), and (owes car) because trunk is a misspelling of truck, motorcycle was not declared to be a VEHICLE, and "owes" is an undeclared predicate, respectively. Syntactic predicate checking is extremely important when a practical system is being developed.

DUCK supports a good expert system debugging environment. There are special editors for modifing atomic assertions, theorems, and LISP functions. There is a workspace manager to help users save files containing modified functions and theorems. There is general graph walking facility that allows users to traverse proofs and data dependency networks. It is possible to ask "why" any result is believed. This is useful in searching for an incorrect theorem that allowed a false result to be generated. If a result cannot be proven, a user can ask the system "why not" and be supplied with theorems that could have been used to generate the result plus the individual subgoals associated with those theorems that were blocked. This enables the user to find and edit the theorem that was

too restrictive or write a new theorem to cover the special case. The "why" and "why not" commands in conjunction with the graph walker are powerful debugging aids.

III. The Structure of CALAP

The FORTRAN version of CALAP displays two different kinds of structure. The first kind of structure is an artifact of implementing the system on a small outdated computer with "overlay" memory management rather than virtual memory management. The module structure of CALAP is defined by the separate modules CALAP, XCAPI, EBSEC, REGA, REGAS, and DETA that call each other as overlays. Figure 1. illustrates this overlay structure.

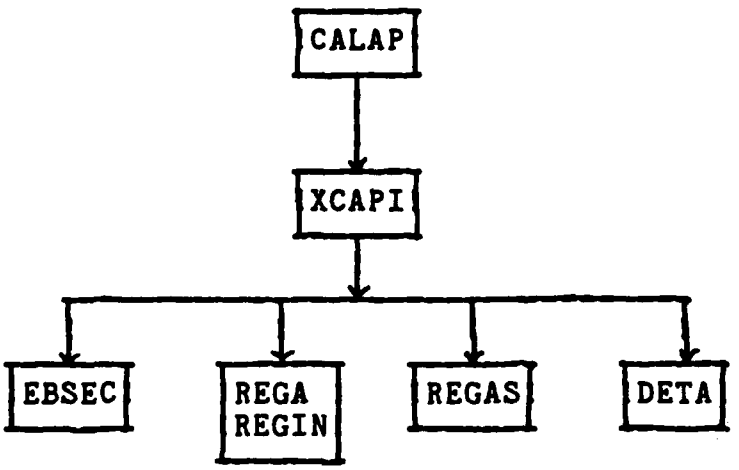


Figure 1: Overlay Structure of CALAP

The second kind of structure describes "what" CALAP does, rather than "how". Figure 2. shows the functional (real) structure of CALAP.

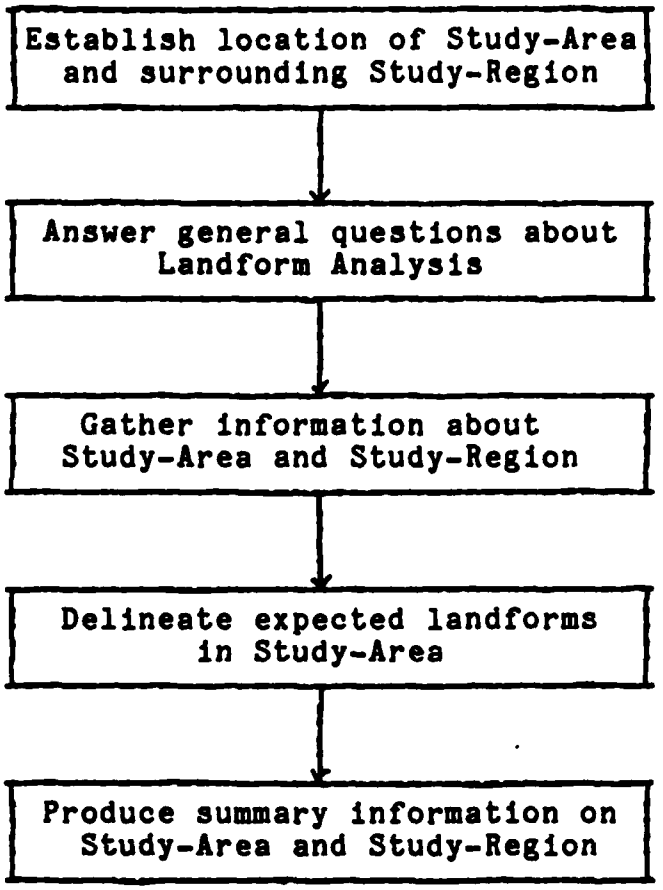


Figure 2: Structure of CALAP

In general, a task can be described procedurally or non-procedurally. A non-procedural description explains "what" is to happen. A procedural description explains "how" a process occurs or "when" things are expected to occur. A pure logic programming description of CALAP would be non-procedural description. The DUCK CALAP system that was written is both a procedural and a non-procedural description. The theorems or rules are defined and called in a non-procedural way. The consequents of some theorems, however, are LISP functions that perform tasks procedurally.

The FORTRAN version of CALAP is a completely procedural description. The OPS5 version of CALAP appears to be a non-procedural description, but in reality the knowledge of the "conflict resolution strategy" leads OPS5 programmers to construct very procedural systems.

FORTRAN is sequential programming language. Unless one constructs a control program, (that essentially mimics the LISP environment), the intelligence of the program lies in the order of the program statements, as the expert wrote it. If the program is implementing a decision tree, then this tree must be linearized, often with great harm to the understandibility of the program. Such a program may be very fast since the "intelligence" is hard coded in-place. On the other hand, such systems are often hard to maintain or improve because the knowledge is not stored modularly, but can be spread all over the program.

IV. DUCK Programming

DUCK, as other logic programming languages, actively encourages modular program development. The basic way a DUCK system works is that a set of predicates, that describe the initial conditions, are asserted into a database. Theorems representing knowledge about the domain also reside in the database. A control program, usually written in LISP, controls the processing. It asserts facts into the database and fetches information from the database. It uses the fetched information to determine the appropriate processing. The modification of the database by the control program and theorems allows the system to transform its understanding of the problem. The control program or theorems with LISP bodies can request more information from the user and assert the responses into the database to further the process of understanding. When the desired result has been generated, the system stops. The user can obtain an explanation of the result.

In any real task, there exist some subtasks that can be performed in parallel or are conditionally performed and other subtasks that must be performed sequentially. As you would expect, DUCK is an excellent vehicle for the parallel or conditional subtasks while FORTRAN does well with the sequential. Since DUCK allows theorems to call LISP functions and LISP functions to perform deductions, DUCK also performs sequential tasks well by using LISP constructs.

Now some of the standard tools used to construct a DUCK system will be described.

A legal action for a theorem is to call a LISP function. This LISP function can create new assertions, or interrogate knowledge-bases, or perform specialized input or output functions. In the case of DUCK CALAP, special routines have been written for user interface. Many DUCK CALAP questions have only yes/no answers. Initially, a question such as "are there tidal rivers in the study area?" will not have been asked and DUCK CALAP represents this by the absence of an (answer s-a-tidal-river t/nil) assertion in the database. DUCK's non-monotonic features are used to determine the absence of the predicate. There is a LISP routine name "user" that asserts (answer s-a-tidal-river t) if the user's response to the question was any of yes, y, t, true, ok, etc. Otherwise, (answer s-a-tidal-river nil) would be asserted. An answer predicate will have no justifiers, if the user supplied the answer. Otherwise, the predicate will have the name of the theorem that asserted it as a justifier.

Another LISP function named "display" allows list representations of slightly modified FORTRAN format statements to be printed out at the terminal. Since so much of CALAP's function is to supply information to the user, this LISP function greatly simplified the CALAP conversion. Note also that these messages are an example of a database of knowledge that is not stored in the assertional database.

DUCK is well suited for conditional information flows, i.e. if this is true, then that is true; if this and that is true, then ... is true (or false), etc. You just create theorems with the necessary antecedents and consequents. What if you have three simple if-then statements that must be executed in a particular order? It's simple in FORTRAN. In DUCK, you would use a LISPRULE instead of a RULE. A RULE statement asserts a theorem into the DUCK database. A LISPRULE statement asserts a theorem whose body is LISP code into the DUCK database. The LISP body can assert facts and theorems into the DUCK database in any necessary order. It can also access the DUCK database and other databases to determine what to do. Likewise, looping and other forms of sequential behavior can be implemented.

Looping, conditional sequencing, and sequential control structures were used extensively during the implementation of DUCK CALAP. For example, DUCK CALAP has a number of menu loops where the user is shown a menu and can choose which item he would like to see next. The user can continue to get information until he types the item that stops the loop. An example of a menu loop, a simplification of DUCK code that occurs in file ebsec.l, follows.

```
(lisprule ebsec-rule (goal ebsec) ->
 (for-first-ans (fetch '(study-region ?s ?p))
   (display 'ebsec-kickoff-msg (list ?s ?p))
   )
 (prog [a1 a2]
   top (for-first-ans (fetch '(study-region ?s ?p))
         (display 'ebsec-menu-msg (list ?s ?p))
         )
       (:= a1 (read))
       (cond [(memq a1 '(1 p physiography phy pg))
                (display 'ebsec-physiography-msg nil)]
             [(memq a1 '(2 lf landforms))
                (display 'ebsec-landform-msg nil)]
             [(memq a1 '(3 g geology s soils geology/soils))
                (display 'ebsec-geology-msg nil)]

               ...

             [(memq a1 '(9 e exit))
                (return (premiss '(done stop)))]
             [(memq a1 '(0 ok)) (return)]
             [t (msg t "Illegal Answer, "
                       "Please try again" t)
                (go top)]
             )
  more (display 'ebsec-more-msg nil)
       (:= a2 (read))
       (cond [(memq a2 '(1 detail more m d))
                (display 'ebsec-physiography-more-msg nil)
                (go top)]
             [(memq a2 '(2 another a)) (go top)]
             [(memq a2 '(3 continue c go g go-on))
                (return)]
             [t (msg t "Sorry: " a2
                       " is not a legal response"
                     t "Please respond properly" t)
                (go more)]
             )
       )
 (premiss '(done ebsec))
 )

(setq ebsec-kickoff-msg
 '(t t 10 "Your study area is in the " (o 1) " of" t
   5 "the " (o 2) " Physiographic Province. The" t
   5 "following menu will provide regional background" t
   5 "information on this physiographic section if you" t
   5 "desire." t t))
```

```
(setq ebsec-menu-msg
 '(t " ****************************************" t
    5 "Do you want information on any of" t t
   10 "1 - Physiography" t
   10 "2 - Landforms" t
   10 "3 - etc." t

      ...

   10 "9 - ****Terminate Program****" t
   10 "0 - Complete regional background phase" t t
     " ****************************************" t t
    5 "Please type your selection number" t t))

(setq ebsec-physiography-msg
 '(t t 5 "PHYSIOGRAPHY OF THE EMBAYED SECTION," t
    5 "ATLANTIC COASTAL PLAIN PHYSIOGRAPHIC PROVINCE" t t
   10 "The Embayed Section of the Atlantic Coastal Plain" t
   10 "a submaturely dissected and partly submerged," t
   10 "terraced coastal plain." t t))

(setq ebsec-more-msg
 '(t t 5 "Type:" t
   10 "1 - For additional information in this category" t
   10 "2 - For regional background information in a" t
   10 "     different category" t
   10 "3 - To complete your regional background" t
   10 "     information phase and proceed with regional" t
   10 "     analysis" t t))
```

Further examples can be found in the Appendix, which contains a copy of DUCK CALAP.

V. Computer Explanation

In [2], two different schools of thought about explanation in expert systems were presented. The opinions of the schools can be summarized.

One school believes that predicate calculus should be the basic knowledge representation language, that the conclusions of an expert system should be logically deduced by inference, and that the an adequate explanation of a conclusion is its proof (i.e. the ordered list of facts (atomic assertions) and rules (theorems) used to prove the conclusion). This kind of explanation facility is general, because it works for any conclusion in any domain. Note that the explanation is in the vocabulary of the system designer or expert, rather than the system user. See McDermott and Brooks [11] for an example of such a system.

The other school of explanation believes that performing expertly is very different from explaining expert behavior. See McDermott [12]. The argument is that the process of adequately explaining the actions, reasons, etc. of an expert system is a task of roughly the same magnitude as constructing the original expert system. Adequate explanation means anticipating the questions a user could ask at different points and understanding what information the user really wants to know. How this information is presented may depend on a model of the user's understanding of the system or on other questions he has asked the system.

In this school, explanation is considered an expert system in itself, requiring all the knowledge engineering tools that the original expert system found useful.

There are two different kinds of explanation, user independent and user dependent. Predicate calculus based systems at least have user independent explanation. Many other systems do not even have that. If one chooses to implement user dependent explanation, then it can be added to predicate calculus based expert systems with roughly the same amount of effort as to non-predicate calculus based (e.g. OPS5) expert systems.

User dependent explanation seldom helps a domain expert or knowledge engineer modify or debug an expert system. It is easier to debug a predicate calculus expert system because one has access to the proof. DUCK, for example, has a walk mode that allows users to traverse the proof tree. By typing "why," DUCK will show why a result is believed. This can be used to search for an incorrect theorem that allows false results to be generated. By typing "whynot," DUCK will show why a result could not be proven. The theorems that could have generated the result (and the individual subgoals associated with those theorems that were blocked) will be printed. This enables the user to find and edit the theorem or theorems that were overly restrictive or create a new theorem that covers the special case.

CALAP is actually a tutorial program, rather than strictly an expert system. CALAP's main function is to provide enough structure so a novice can delineate landforms in an image with the help of a map. CALAP accomplishes this end by offering and supplying information and help to the user. CALAP is a user dependent explanation system that allows expert users to skip over unnecessary explanations but allows novices to acquire both general and specific explanations.

It is interesting to note that FORTRAN CALAP runs very well but has no knowledge of or representation for its actions. OPS5 CALAP could leave a representation of its actions in working memory, but does not. Many of the actions of DUCK CALAP are recorded by DUCK and can be displayed by walking through the various proofs generated.

VI. Conclusion

It is well known that, barring memory limitations, real-time constraints, etc., any task can be made to work in almost any programming language. The relevent question is which languages provide good structure and environment for achieving a task. We will discuss the advantages and disadvantages of implementing CALAP in FORTRAN, LISP, OPS5, and DUCK.

In the case of CALAP, FORTRAN is the wrong language because: 1) it has forced a linearization of (introduced artificial structure in) CALAP that is difficult to understand; 2) it has variable name conventions that prevent the use of meaningful variable names; and 3) it does not allow the chunks of knowledge to be stored modularly but instead places it only where needed (efficient but hard to understand or maintain).

LISP is a much better language for developing CALAP because it does not suffer from FORTRAN weaknesses 1) and 2). In particular, the artificial linearization could be eliminated because the graph or tree structure that represents CALAP could be easily constructed and manipulated in LISP. However, LISP too is mainly a sequential language and if knowledge modularity is not enforced by use of some kind of knowledge database, then a LISP implementation could have FORTRAN weakness 3). The other advantages of LISP are: 1) its powerful and flexible data structures are easy to use

and do not require user storage allocation; 2) its powerful online editing, tracing, and debugging environment; and 3) its support of recursive programming, if needed.

OPS5 appears to be much better than FORTRAN and it is probably better than LISP for implementing CALAP. It does not have FORTRAN weaknesses 1), 2), or 3), but does require the introduction of artificial structure to implement sequential production sequencing or looping. OPS5 allows a production to call LISP functions. This escape hatch allows databases of knowledge to be accessed without filling up working memory, patently sequential tasks to be performed in LISP, special input/output function to be called, etc. OPS5 has all the other advantages of the LISP environment, data structures, debugging aids, and recursion, etc. Two improvements to OPS5 would be: 1) allowing structure in working memory elements and, 2) allowing production sequencing and looping without the introduction of artificial control structure.

DUCK appears to be much better than FORTRAN, LISP, or OPS5 for implementing CALAP. It does not have FORTRAN weaknesses 1), 2), or 3). It is LISP with the addition of many useful capabilities. It does not require the introduction of artificial structures to implement procedural tasks (as OPS5 does) because it allows theorems to call LISP functions which themselves can cause further logic operations, etc. This escape hatch allows databases of knowledge to be accessed without cluttering up the

assertion database, patently sequential tasks to be performed in LISP (augmented by DUCK), special input/output function to be called, etc. DUCK has all the other advantages of the LISP environment, data structures, debugging aids, and recursion, etc. One has both the benefits of a non-procedural description and the ability choose a procedural description when efficiency or common sense demands it. DUCK implements both forward and backward chaining, multiple databases, non-monotonic operations, a reason maintainence system, common sense rules, automatic syntactic checking of predicates, a rudimentary explanation facility, and a powerful logic debugging environment.

Now there are three different versions of CALAP. Both the OPS5 and the DUCK were written to be alike so they could be easily compared with each other and the original FORTRAN version. It is not surprising that CALAP could be implemented in DUCK, since DUCK has so many features. The DUCK version seems to be the most transparent and understandible implementation. In [2] we concluded that the OPS5 version of CALAP was somewhat clearer than the original FORTRAN version.

A DUCK version of CALAP has been implemented using approximately 32 theorems. Most of these theorems have consequents that are written in LISP. The OPS5 version of CALAP required approximately 220 productions. The granularity, (amount of knowledge in each rule), of the DUCK version of CALAP is much larger than that of the OPS5

version. DUCK CALAP would probably be easier to understand if its granularity was smaller: using perhaps 70 theorems. The DUCK version of CALAP was much easier to program than the OPS5 version. This was due, in large part, to the ease with which the general non-procedural nature of predicate calculus could be subverted in DUCK when clearly procedural tasks needed to be performed. This enabled DUCK CALAP to exploit the advantages of LOGIC and LISP while avoiding the drawbacks.

The current theories about explanation in expert systems have been presented. CALAP is really a tutorial system. The advantages and disadvantages of implementing CALAP in several different languages have been considered. The advantages of a logic programming approach are that the knowledge in the system can be represented in a clear, concise, and uniform manner; the underlying reasoning of the system is logical (i.e. has a proof); and the system can implement an effective explanation mode by allowing proofs to be manipulated and displayed (this also creates an efficient debugging environment). These advantages stem from a modular approach that allows people to quickly prototype systems. The disadvantages are the space and time costs involved in using (accessing, updating, etc.) a uniform knowledge representation instead of a more specialized and efficient representation. These disadvantages are the cost of implementing a modular approach.

VII. References

1. McDermott, Drew, DUCK: A Lisp-Based Deductive System, Department of Computer Science, Yale University, May 1983, available from Smart Systems Technology, 6870 Elm Street, McLean, Virginia, 22101.

2. Hayes, Kenneth C., Convertion of the CALAP Program from FORTRAN to OPS5, Final Report to US Army Engineer Topographic Laboratory, prepared by Smart Systems Technology, Inc. under contract to JAYCOR.

3. Forgy, Charles L., OPS5 User's Manual, CMU Technical Report CMU-CS-81-135, Carnegie-Mellon University, Pittsburgh, Pennsylvania, July 1981.

4. Frege, G., Begriffsschrift, a formula language modelled upon that of arithmetic, for pure thought. In J. van Heijenoort (Ed.), From Frege to Godel: A Source Book In Mathematical Logic, 1879-1931, Harvard University Press, Cambridge, Mass., 1967, 1-81.

5. Robinson, J. Alan, A Machine-oriented Logic Based on the Resolution Principle, JACM, 12(1), 1965, 23-41.

6. Kowalski, R., Predicate Logic as a Programming Language, Information Processing 74, North-Holland, Amsterdam, Holland, 1974, 569-574.

7. Sussman, G. J. and McDermott, Drew V., From PLANNER to CONNIVER--A Genetic Approach, Proc. FJCC, Vol. 41, 1972, 1171.

8. Robinson, J. Alan and Sibert, E. E., Logic Programming in LISP, Technical Report 8-80, Syracuse University, Syracuse, New York, December 1980.

9. Roussel, P., PROLOG: Manual de Reference et D'utilisation, Groupe d'Intelligence Artificielle, Marseille-Luminy, France, September 1975.

10. Warren, D. H. D., Logic Programming and Compiler Writing, Research Report No. 44, Dept. of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland, 1977.

11. McDermott, Drew and Brooks, Ruven, ARBY: Diagnosis with Shallow Causal Models, Proc. AAAI Conference, Pittsburgh, Pennsylvania, August 1982.

12. McDermott, John, XSEL: A Computer Salesperson's Assistant, CMU Technical Report, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1982.

VIII. Appendix A: DUCK CALAP

The source code listings for the DUCK CALAP system appear in this appendix. Much of the original FORTRAN code and all the OPS5 productions remain as comments for the DUCK logic program. The subsections are labelled lit.l, lispcode.l, sys.l, calap.l, xcapi.l, ebsec.l, rega.l, regin.l, regas.l, and deta.l.

```
;
;       lit.l           OPS5 Literalize Definitions
;       Kenneth Hayes   Smart Systems Technology
;
(defducktype STATE SYMBOL)
(duclare STATE
         begin calap
         xcapi xcapi1 xcapi1-y xcapi1-n xcapi2 xcapi2-y xcapi2-n
         ebsec regin
         rega ocean-tidal-river ntriver triver low-sa-max low-sr-max
              low-ter mid-ter high-ter
              drainage special listing
         regas regas-coastal regas-tidal-river
         deta shore shore-menu tidal-river tidal-menu
         stop
         )
;
(defducktype DEG NUMBER)  ;should be between 0 and 60
;
(defducktype CORNER SYMBOL)
(duclare CORNER northwest-corner southeast-corner)
;
(defducktype SECTION SYMBOL)
(duclare SECTION Embayed-Section)
;
(defducktype PLAIN SYMBOL)
(duclare PLAIN Atlantic-Coastal-Plain)
;
(defducktype FEATURE SYMBOL)
(duclare FEATURE
         srmax srmin samax samin
         s-r-coastal-shoreline
         s-r-coastal-shoreline-ext
         s-r-tidal-river
         s-r-tidal-river-ext
         s-r-tidal-river-islands
         s-r-low-coastal-shoreline
         s-r-low-tidal-river
         s-r-low-alluvial-channel
         s-r-tidal-river-rock
         s-r-tidal-river-mud
         s-r-tidal-river-sand
         s-r-low-terraces
         s-r-low-terraces-near-coast
         s-r-low-terraces-near-tidal-river
         s-r-mid-terraces
         s-r-high-terraces
         s-r-rivers
         s-r-meandering-rivers
         s-r-low-meandering-rivers
         s-r-high-meandering-rivers
         s-r-rapid/fall-rivers
         s-r-deflected-rivers
         s-r-high-deflected-rivers
         s-r-streams
         s-r-meandering-streams
         s-r-high-meandering-streams
         s-r-distributary-segments
         s-r-high-distributary-segments
```

```
            s-r-strange-gradients
            s-r-strange-gradients-incised
            s-a-coastal-shoreline
            s-a-coastal-shoreline-ext
            s-a-tidal-river
            s-a-tidal-river-ext
            s-a-tidal-river-islands
            s-a-low-coastal-shoreline
            s-a-low-tidal-river
            s-a-low-alluvial-channel
            s-a-max-coastal-shoreline
            s-a-max-tidal-river
            s-a-max-un-associated
            s-a-tidal-river-rock
            s-a-tidal-river-mud
            s-a-tidal-river-sand
            s-a-low-terraces
            s-a-low-terraces-near-coast
            s-a-low-terraces-near-tidal-river
            s-a-mid-terraces
            s-a-high-terraces
            s-a-rivers
            s-a-meandering-rivers
            s-a-low-meandering-rivers
            s-a-high-meandering-rivers
            s-a-rapid/fall-rivers
            s-a-deflected-rivers
            s-a-high-deflected-rivers
            s-a-streams
            s-a-meandering-streams
            s-a-high-meandering-streams
            s-a-distributary-segments
            s-a-high-distributary-segments
            s-a-strange-gradients
            s-a-strange-gradients-incised
            s-r-gravel-pits
            s-r-quarry
            s-a-coastal-beach
            s-a-coastal-beach-ridge
            s-a-coastal-swale
            s-a-coastal-sand
            s-a-coastal-ponded-depression
            s-a-coastal-islands
            s-a-coastal-bars
            s-a-coastal-bed-rock
            s-a-gravel-pits
            s-a-quarry
            )
;
(defpred (goal STATE ?s))
;
(defpred (done STATE ?s))
;
(defpred (nextgoal STATE ?prev STATE ?new))
;
(defpred (gcoord CORNER ?place
                 DEG ?northdegree DEG ?northminute DEG ?northsecond
                 DEG ?eastdegree DEG ?eastminute DEG ?eastsecond))
;
```

```
(defpred (study-region SECTION ?s PLAIN ?p))
;
(defpred (feature FEATURE ?f))
;
(defpred (spfeature FEATURE ?f))
;
(defpred (answer FEATURE ?f OBJ ?answer))
;
(defpred (spanswer FEATURE ?f (1st NUMBER) ?answer))
;
(defpred (unasked FEATURE ?f))
;
(defpred (ansed FEATURE ?f))
;
(defpred (ask FEATURE ?f OBJ ?q OBJ ?c))
;
(defpred (ask-n FEATURE ?f OBJ ?q OBJ ?c))
;
(defpred (do FEATURE ?f OBJ ?c))
;
;(literalize northwest-corner nd nm ns ed em es)
;(literalize southeast-corner nd nm ns ed em es)
;(literalize study-region section plain)
;(literalize cstack cp ps)
;(literalize push   cp ps)
;(vector-attribute ps)
(princ "LIT.L Loaded")
(terpr)
```

LISPCODE.L

```
;
;       lispcode.l      LISP Support Functions
;       Kenneth Hayes   Smart Systems Technology
;
;---
;     (display format args)  - FORTRAN-like formatted printing
;
(defun display (format args)
  (cond [(symbolp format) (:= format (eval format))])
  (do [(l format (cdr l))]
      [(null l) nil]
      (disprint (car l) args)
      )
  )
;
(defun disprint (item args)
  (cond [(stringp item) (princ item)]
        [(numberp item) (dispaces item)]
        [(eq item t) (terpr)]
        [(and (consp item) (eq (car item) 'e)) (print (cadr item))]
        [(and (consp item) (eq (car item) 'o))
           (print (nthelem (cadr item) args))]
        [t (print item)])
  )
;
;      (display-tf format args) - FORTRAN-like formatted printing
;                                 for logical variables
;
(defun display-tf (format args)
  (cond [(symbolp format) (:= format (eval format))])
  (do [(l format (cdr l))]
      [(null l) nil]
      (disprinttf (car l) args)
      )
  )
;
(defun disprinttf (item args)
  (cond [(stringp item) (princ item)]
        [(numberp item) (dispaces item)]
        [(eq item t) (terpr)]
        [(and (consp item) (eq (car item) 'e)) (print (cadr item))]
        [(and (consp item) (eq (car item) 'o))
           (setq item (nthelem (cadr item) args))
           (cond [(eq t item) (princ "T")]
                 [item (princ item)]
                 [t (princ "F")])]
        [t (print item)])
  )
;
;      (display-and format test args) - conditional FORTRAN-like
;                                       formatted printing
;
(defun display-and (format flag args)
  (cond [(symbolp format) (:= format (eval format))])
  (cond [flag (do [(l format (cdr l))]
                  [(null l) nil]
                  (disprint (car l) args)
                  )])
  )
```

```
;
(defun dispaces (n)
  (do [(i 1 (1+ i))]
      [(greaterp i n)]
      (princ " ")
      )
  )
;
;---
;       (pause) - causes a wait until the user types a <cr>.
;
(declare (special nopause*))
(defun pause ()
  (cond [nopause*]
        [t (princ "PAUSEing: Type <cr> to continue")
           (do [(c (tyi) (tyi))]
               [(eq c 10)]
               (and (null c)
                    (terpr)
                    (princ "Error in PAUSE")
                    (return))
               )])
  )
;
(setq nopause* nil)
;
;---
;       (dread) - Does a standard read and eats the <cr>
;                 that follows the input.
;
(defun dread ()
  (prog1 (read) (readcr))
  )
;
;       (readcr) - reads up to the next <cr>.
;
(defun readcr ()
  (do [(c (tyi) (tyi))]
      [(eq c 10)]
      (cond [(null c) (return)])
      )
  )
;
(putd 'r (getd 'reset))
;
;;---
;;
;(defun display () (ops-prints (eval ($parameter 1)) 1))
;;
;;---
;;
;(defun display-tf () (ops-prints-tf (eval ($parameter 1)) 1))
;;
;;---
;;
;(defun display-and () (ops-prints-and (eval ($parameter 1)) 1))
;;
;;---
;;
```

```
;(defun cond-display ()
;     (cond [(null ($parameter 1))]
;           [t (ops-prints (eval ($parameter 2)) 2)])
;      )
;;
;;---
;;
;(defun asked-display ()
;     (cond [(eq ($parameter 1) 'unasked)]
;           [(null ($parameter 1))]
;           [t (ops-prints (eval ($parameter 2)) 2)])
;      )
;;
;;
;(defun ops-prints (format offset)
;     (cond [(null format) nil]
;           [t (ops-print (car format) offset)
;              (ops-prints (cdr format) offset)])
;      )
;;
;(defun ops-prints-tf (format offset)
;     (cond [(null format) nil]
;           [t (ops-print-tf (car format) offset)
;              (ops-prints-tf (cdr format) offset)])
;      )
;;
;(defun ops-prints-and (format offset)
;     (cond [(null format) nil]
;           [(ops-print-and (car format) offset)
;              (ops-prints-and (cdr format) offset)])
;      )
;;
;;
;(defun ops-print (item offset)
;     (cond [(stringp item) (princ item)]
;           [(numberp item) (spaces item)]
;           [(eq item t) (terpr)]
;           [(and (consp item) (eq (car item) 'e)) (print (cadr item))]
;           [(and (consp item) (eq (car item) 'o))
;              (print ($parameter (+ (cadr item) offset)))]
;           [t (print item)])
;      )
;;
;(defun ops-print-tf (item offset)
;     (cond [(stringp item) (princ item)]
;           [(numberp item) (spaces item)]
;           [(eq item t) (terpr)]
;           [(and (consp item) (eq (car item) 'e)) (print (cadr item))]
;           [(and (consp item) (eq (car item) 'o))
;              (setq item ($parameter (+ (cadr item) offset)))
;              (cond [(eq t item) (princ "T")]
;                    [item (print item)]
;                    [t (princ "F")])]
;           [t (print item)])
;      )
;;
;(defun ops-print-and (item offset)
;     (cond [(stringp item) (princ item) t]
;           [(numberp item) (spaces item) t]
```

```
;           [(eq item t) (terpr) t]
;           [(and (consp item) (eq (car item) 'e)) (print (cadr item)) t]
;           [(and (consp item) (eq (car item) 'o))
;              (cond [(setq item ($parameter (+ (cadr item) offset)))
;                        (print item) t]
;                    [t nil])]
;           [t (print item) t])
;      )
;;
;;
;(defun spaces (n)
;     (do [(i 1 (1+ i))]
;         [(greaterp i n)]
;         (princ " ")
;         )
;      )
;;
;;---
;;
;(defun clear () (zapline))
;;
;;---
;;
;(defun yes-no ()
;     ($tab 2)
;     ($value
;       (cond [(memq (read-first)
;                    '(Y y YES yes Yes
;                      OK ok Ok TRUE true T t GO go Go)) t]
;             [t nil]))
;     ($value 'user)
;     ($assert)
;      )
;;
;;---
;;
;(defun input1 ()
;     ($value (read-first))
;     ($assert)
;      )
;;
;(defun read-first ()
;     (do [(c (tyi) (tyi)) (l nil)]
;         [(equal 10 c) (apply 'concat (nreverse l))]
;         (setq l (cons (ascii c) l))
;         )
;      )
;;
;;---
;;
;(declare (special nopause*))
;(defun pause ()
;     (cond [nopause*]
;           [t (princ "PAUSEing: Type <cr> to continue")
;              (do [(c (tyi) (tyi))]
;                  [(eq c 10)]
;                  (and (null c)
;                       (terpr)
;                       (princ "Error in PAUSE")
```

```
;                          (return))
;                  )])
;       )
;
;(setq nopause* nil)
;;
;(putd 'consp (getd 'dtpr))
;;
;(putd 'r (getd 'reset))
;;
(princ "LISPCODE.L Loaded")
(terpr)
```

SYS.L

```
;
;       sys.l           Control and Utility Routines
;       Kenneth Hayes   Smart Systems Technology
;
;       (calap)
;
;       CALAP is the main control program of DUCK CALAP.
; CALAP runs in a private data-pool.
;
(defun calap ()
  (let [(dp* (dp-push dp*))]
       (do [(over nil)
            (waste (premiss '(done begin)))
            (ii 1 (add1 ii))]
           [over (progn (msg t "CALAP Signing off" t)
                        (duck)
                        (dp-kill dp*))]
           (cond [calap-dbg* (msg t "CONTROL LOOP " ii t)
                             (duck)
                             (readcr)])
           (for-first-ans (fetch '(done stop))
             (:= over t)
             )
           (for-first-ans (fetch '(and (done ?x)
                                       (nextgoal ?x ?y)
                                       (thnot (goal ?y))))
             (cond [over]
                   [(eq ?y 'stop) (:= over t)]
                   [t (msg t "CALAP STEP " ?y t)
                      (errset (premiss `(goal ?y)) nil)])
             )
           )
       )
  )
;
(setq calap-dbg* nil)
;
;       (gans feat) - returns the answer associated with feature
;                     FEAT, or NIL if there is no answer.
;
(defun gans macro (exp)
  `(let [(x (assertion-fetch '(answer ,(cadr exp) ?x)))]
        (cond [x ?(x (car x))])
        )
  )
;
;       (gans* feat) - returns the answer associated with feature
;                      FEAT, or NIL if there is no answer. FEAT
;                      is evaluated.
;
(defun gans* (feat)
  (:= feat (assertion-fetch `(answer ,feat ?x)))
  (cond [feat ?(x (car feat))])
  )
;
;       (gans+ feat) - returns the answer associated with feature
;                      FEAT, or "?" if there is no answer.
;
(defun gans+ macro (exp)
```

```
  `(let [(x (assertion-fetch '(answer ,(cadr exp) ?x)))]
        (cond [x ?(x (car x))]
              [t "?"])
        )
  )
;
;       (gansp feat) - returns the "special" answer associated with
;                      feature FEAT, or NIL if the is no special
;                      answer. GANSP is only used for the gravel-pit
;                      and quarry features.
;
(defun gansp macro (exp)
  `(let [(x (assertion-fetch '(spanswer ,(cadr exp) ?x)))]
        (cond [x ?(x (car x))])
        )
  )
;
;       (unans feat) - returns NIL if there is an answer associated
;                      with FEAT, otherwise, non-NIL is returned.
;
(defun unans macro (exp)
  `(assertion-fetch '(unasked ,(cadr exp)))
  )
;
;       (unans* feat) - returns NIL if ther is an answer associated
;                       with FEAT, otherwise, non-NIL is returned.
;                       FEAT is evaluated.
;
(defun unans* (feat)
  (assertion-fetch `(unasked ,feat))
  )
;
;       (user feat ques)
;
;       USER returns the value of feature FEAT. If FEAT has not been
; answered, then print out QUES and assert the user's response as
; the answer to feature FEAT.
;
(defun user (feat ques)
  (cond [(unans* feat)
           (display ques nil)
           (:= ques (and (is-yes (dread)) t))
           (premiss `(answer ,feat ,ques))]
        [t (:= ques (gans* feat))])
  ques
  )
;
;       (ask feat question msg)
;
;       ASK causes an answer to be associated with feature FEAT, if
; no answer is currently associated with FEAT. QUESTION is what is
; printed to the user before the answer is read. If the answer
; associated with FEAT is T and MSG is non-NIL, then the predicate
; (do FEAT MSG) is asserted into the database.
;
(lisprule askit (ask ?feat ?ques ?cmd) ->
  (user ?feat ?ques)
  (cond [?cmd (premiss '(-> (answer ?feat t) (do ?feat ?cmd)))])
  )
```

```
;
;       (ask-n feat question feat2)
;
;       ASK-N causes an answer to be associated with feature FEAT, if
; no answer is currently associated with FEAT. QUESTION is what is
; printed to the user before the answer is read. If the answer
; associated with FEAT is NIL, then the answer NIL is associated
; with feature FEAT2.
;
(lisprule ask-nil-prop (ask-n ?feat1 ?msg ?feat2) ->
  (user ?feat1 ?msg)
  (cond [(not (gans* ?feat1))
         (premiss '(answer ?feat2 nil))])
  )
;
;       (do feat msg)
;
;       DO causes message MSG to be printed to the user and DUCK
; to pause until the user types a <cr>.
;
(lisprule doit (do ?feat ?cmd) ->
  (display ?cmd nil)
  (pause)
  )
;
;(external display)
;(external display-tf)
;(external display-and)
;(external cond-display)
;(external asked-display)
;(external clear)
;(external input1)
;(external pause)
;(external yes-no)
;;
;(p start-rule
;   (start)  -->
;     (remove 1)
;     (make done start)
;     (make cstack ^cp start ^ps calap xcapi alldone)
;     )
;;
;(p main-control-rule
;   (done <a>)
;   (cstack ^cp <a> ^ps <b>) -->
;     (modify 1 goal (substr 2 3 3))
;     (remove 2)
;     (make cstack ^cp (substr 2 3 3) ^ps (substr 2 4 inf))
;     )
;;
;(p push-control-rule
;   (goal <a>)
;   (push ^cp <a>)
;   (cstack ^cp <a>) -->
;     (modify 1 ^2 (substr 2 3 3))
;     (remove 2)
;     (modify 3 ^cp (substr 2 3 3)
;               ^ps (substr 2 4 inf) (substr 3 3 inf))
;     )
```

```
;;
;(p alldone-rule
;    (goal alldone) -->
;      (remove 1)
;      (write (crlf) "CALAP all done" (crlf))
;      (halt)
;      )
;;
(terpr)
(princ "SYS.L Loaded")
(terpr)
```

CALAP.L

```
;
;       calap.l     Computer-Assisted Landform Analysis Program
;
;       Fortran System:   Robert Leighty    USAETL
;       OPS5 System:      Kenneth Hayes     Smart Systems Technology
;       DUCK System:      Kenneth Hayes     Smart Systems Technology
;
;;      PROGRAM CALAP
;;
;;  ************************************************************************
;;
;;   COMPUTER-ASSISTED LANDFORM ANALYSIS PROGRAM
;;
;;   AN INTERACTIVE PROGRAM WHICH LEADS A PHOTO INTERPRETER THROUGH
;;   A LANDFORM ANALYSIS PROBLEM FOR ANY STUDY AREA IN A SELECTED
;;   PHYSIOGRAPHIC SECTION.
;;
;;   RDL, MAY 81
;;
;;  ************************************************************************
;;
;;
;;
;;      DIMENSION ICALAP(3),IYES(2)
;;
;;      DATA ICALAP/2HXC,2HAP,2HI /
;;      DATA IYES/2HYE,2HS /
;;
;;
;;      CALL RMPAR(LU)
;;      IF(LU(1).EQ.0)LU(1)=1
;;      ISTEP=1
;;
;;  ------------------------------------------------------------------
;;   SYSTEM INTRODUCTION
;;  ------------------------------------------------------------------
;;
;
(lisprule calap-rule (goal calap) ->
  (display 'calap-start-msg nil)
  (pause)
  (prog [ans1 ans2]
    top (display 'calap-coord-msg nil)
        (:= ans1 (dread))
        (cond [(eq ans1 'help)
                  (display 'calap-coord-help nil)
                  (go top)]
              [(eq ans1 'ok)
                  (:= ans1 '(38 41 30 77 10 25))
                  (:= ans2 '(38 40 45 77  9  5))
                  (premiss `(gcoord northwest-corner .,ans1))
                  (premiss `(gcoord southeast-corner .,ans1))
                  (go chk)]
              [(eq ans1 'exit) (return (premiss '(done stop)))]
              [(and (consp ans1) (equal (length ans1) 6))
                  (premiss `(gcoord northwest-corner .,ans1))]
              [t (msg t "Improper Input, Please try again" t)
                  (go top)]
              )
```

```
   get2 (:= ans2 (dread))
        (cond [(and (consp ans2) (equal (length ans2) 6))
                   (premiss `(gcoord southeast-corner .,ans2))]
              [t (msg t "Improper Input, Try again with second coordinate"
                 (go get2)]
              )
    chk (display 'calap-coord-display (append ans1 ans2))
        (:= ans1 (dread))
        (cond [(eq ans1 'exit)
                   (return (premiss '(done stop)))]
              [(is-yes ans1)
                 (premiss '(study-region Embayed-Section
                                         Atlantic-Coastal-Plain))]
              [t (display 'calap-back nil)
                 (go top)]
              )
   done (premiss '(done calap))
        (return)
        )
  )
;
;(p calap-rule-1
;   (goal calap) -->
;     (call display calap-start-msg)
;     (make goal calap-input-coord)
;     )
;;
;;       WRITE(LU(2),4)
(setq calap-start-msg
     '(t t t 5 "WELCOME TO CALAP" t t
       10 "The acronym CALAP stands for computer-assisted" t
       5 "landform analysis program." t t
       10 "During the course of the program you will be asked" t
       5 "to enter (type) information via the keyboard." t
       5 "Requests relating to a selection from a menu or a" t
       5 "simple yes/no answer are self explanatory, but where" t
       5 "alphanumeric data is requested, a format will usually" t
       5 "be given." t t
       10 "We will assume you have a 1:50,000 scale topographic" t
       5 "map sheet/sheets containing a rectangular geographic" t
       5 "area in which you wish to identify and delineate the" t
       5 "landforms. Here after this area will be referred" t
       5 "to as the 'study area' and the remainder of" t
       5 "the topographic map sheet will be called the" t
       5 "'study region'. We also will assume that you have " t
       5 "aerial imagery of two photo scales: a small scale" t
       5 "of monoscopic or stereoscopic photos with scales" t
       5 "of <1:80,000 and a stereoscopic set of images with" t
       5 "photo scales of 1:50,000 or larger." t t))
;;
;; ------------------------------------------------------------
;;  STUDY AREA COORDINATE ENTRY
;; ------------------------------------------------------------
;;
;;     5 WRITE(LU(2),6)
(setq calap-coord-msg
     '(t t 10 "Now please enter the latitude-longitude" t
       5 "coordinate in degrees, minutes, and seconds for the" t
       5 "northwest and southeast corners of the rectangular" t
```

```
        5 "study area. (ie. 2 separate lines, each line consists" t
        5 "of 6 #s separated by blanks and surrounded by parentheses)" t t
        5 "Type 'help<cr>' if you do not understand the format" t
        5 "desired, 'ok<cr>' for the standard coordinates," t
        5 "or 'exit<cr>' to exit the program." t ))
;;
;(p calap-input-coord-rule-1
;    (goal calap-input-coord) -->
;      (call display calap-coord-msg)
;      (make northwest-corner (accept))
;      (call clear)
;      )
;;
;(p calap-input-coord-rule-2
;    (goal calap-input-coord) -->
;      (make southeast-corner (accept))
;      (call clear)
;      )
;;
;(p calap-coord-help-rule
;    (goal calap)
;    (goal calap-input-coord)
;    (northwest-corner help) -->
;      (remove 2 3)
;      (make goal calap-input-coord)
;      (call display calap-coord-help)
;      )
;;
;(p calap-coord-ok-rule-1
;    (goal calap)
;    (goal calap-input-coord)
;    (northwest-corner ok) -->
;      (remove 3)
;      (make northwest-corner 38 41 30 77 10 25)
;      (make southeast-corner 38 40 45 77  9  5)
;      )
;;
;(p calap-coord-ok-rule-2
;    (goal calap)
;    (goal calap-input-coord)
;    (northwest-corner ok)
;    (southeast-corner) -->
;      (remove 3 4)
;      (make northwest-corner 38 41 30 77 10 25)
;      (make southeast-corner 38 40 45 77  9  5)
;      )
;;
;(p calap-coord-bye-rule
;    (goal calap)
;    (goal calap-input-coord)
;    (northwest-corner exit) -->
;      (remove 1 2 3)
;      (halt)
;      )
;;
;;     8 READ(LU(1),*)MLAT1(1),MLAT1(2),MLAT1(3),MLON1(1),MLON1(2),
;;      *   MLON1(3),MLAT2(1),MLAT2(2),MLAT2(3),MLON2(1),MLON2(2),MLON2(3)
;;
;;       IF(MLAT1(1).NE.0)GO TO 25
```

```
;;
;;  ---------------------------------------------------------------
;;   STUDY AREA COORDINATE ENTRY ASSISTANCE
;;  ---------------------------------------------------------------
;;
;;       WRITE(LU(2),10)
(setq calap-coord-help
      '(t t " EXAMPLE OF THE DESIRED FORMAT:" t
        5 "If the latitude and longitude, in degrees," t
        5 "minutes, and seconds, of a corner" t
        5 "of the study area is (N 38D 41' 30'', E 77D 10' 25'')" t
        5 "then it should be typed as" t
        5 "(38 41 30 77 10 25)<cr>" t t
        5 "OK let's try it for your coordinates." t t))
;;
;;       GO TO 8
;;
;;    25 IF(MLAT1.EQ.9)GO TO 100
;;
;;  ---------------------------------------------------------------
;;   DISPLAY COORDINATES
;;  ---------------------------------------------------------------
;;
;(p calap-display-rule
;    (goal calap-input-coord)
;    (northwest-corner <a> <b> <c> <d> <e> <f>)
;    (southeast-corner <u> <v> <w> <x> <y> <z>) -->
;      (call display calap-coord-display
;                    <a> <b> <c> <d> <e> <f>
;                    <u> <v> <w> <x> <y> <z>)
;      (make calap (accept))
;      (call clear)
;      )
;;
;;       WRITE(LU(2),30)MLAT1(1),MLAT1(2),MLAT1(3),MLON1(1),MLON1(2),
;;      *  MLON1(3),MLAT2(1),MLAT2(2),MLAT2(3),MLON2(1),MLON2(2),MLON2(3)
(setq calap-coord-display
      '(t " Is your northwest corner coordinate:" t
        5 "N " (o 1) "d " (o 2) "' " (o 3) "'',  "
            "E " (o 4) "d " (o 5) "' " (o 6) "''" t
        " and your southeast corner coordinate:" t
        5 "N " (o 7) "d " (o 8) "' " (o 9) "'',  "
            "E " (o 10) "d " (o 11) "' " (o 12) "''" t t
        " If so type 'yes', if not type 'no'." t
        " If you want to stop, type 'exit'." t t))
;;       READ(LU(1),35)IANS
;;    35 FORMAT(2A2)
;;
;(p calap-done-rule
;    (goal calap)
;    (goal calap-input-coord)
;    (calap << yes YES y Y t T true TRUE ok OK >>) -->
;      (modify 1 ^1 done)
;      (remove 2 3)
;      (make study-region ^section |Embayed Section|
;                         ^plain   |Atlantic Coastal Plain|)
;      )
;;
;(p calap-loop-rule
```

```
;    (goal calap)
;    (goal calap-input-coord)
;    (northwest-corner)
;    (southeast-corner)
;    (calap << no NO n N nil NIL >>) -->
;      (modify 2 ^1 goal)
;      (remove 3 4 5)
;      (call display calap-back)
;      )
;;
;(p calap-gone-rule
;    (goal calap)
;    (goal calap-input-coord)
;    (calap << exit EXIT e E >>) -->
;      (remove 1 2 3)
;      )
;;
;(p calap-whoa-rule
;    (goal calap)
;    (calap <ans>) -->
;      (write (crlf) "Sorry your answer of " <ans>)
;      (write (crlf) "was not 'yes', 'no', or 'exit'. Please try again")
;      (write (crlf))
;      (modify 2 ^2 (accept))
;      (call clear)
;      )
;;
;;       IF(IANS.EQ.IYES)GO TO 50
;;       WRITE(LU(2),40)
(setq calap-back '(t t " OK let's try again!" t t))
;;       GO TO 5
;;    50 CALL EXEC(8,ICALAP)
;;
;;   100 END
(terpr)
(princ "CALAP.L Loaded")
(terpr)
```

XCAPI.L

```
;
;       xcapi.l     System Overlay Program
;
;       Fortran System:   Robert Leighty    USAETL
;       OPS5 System:      Kenneth Hayes     Smart Systems Technology
;       DUCK System:      Kenneth Hayes     Smart Systems Technology
;
;;      PROGRAM XCAPI(5)
;;
;; ***********************************************************************
;;
;;  EXECUTIVE SEGMENT OF PROGRAM CALAP
;;
;;  RDL, MAY 81
;;
;; ***********************************************************************
;;
;;      COMMON LU(5),MLAT1(3),MLON1(3),MLAT2(3),MLON2(3),ISTEP,MINR,
;;     *  MAXR,MINS,MAXS
;;
;;      COMMON  IRC,ISC,IRCE,ISCE,IRTR,ISTR,IRTRE,ISTRE,IRLC,
;;     *  IRLAC,ISLC,ISLTR,ISLAC,ISMC,ISMTR,IRTRR,IRTRM,IRTRS,ISTRR,
;;     *  ISTRM,ISTRS,ISMZ,IRT1O,IRT1C,IRT1TR,IST1O,IST1C,IST1TR,IRT2O,
;;     *  IST2O,IRT3O,IST3O,IRAC,IRACM,IRACML,IRACMH,IRARF,IRACD,IRACDH,
;;     *  IRS,IRSM,IRSMH,IRSD,IRSDH,IRSG,IRSGI,ISAC,ISACM,ISACMH,ISARF,
;;     *  ISACD,ISACDH,ISS,ISSM,ISSDH,ISSG,ISSGI,IRLTR,ISSD,ISSMH,
;;     *  IRGP(20),IRRG(20),ISGP(20),ISRG(20),IRTRI,ISTRI
;;
;;
;;      DIMENSION IEBSEC(3),IREGA(3),IREGAS(3),IDETA(3)
;;
;;      DATA IEBSEC/2HEB,2HSE,2HC /
;;      DATA IREGA/2HRE,2HGA,2H  /
;;      DATA IREGAS/2HRE,2HGA,2HS /
;;      DATA IDETA/2HDE,2HTA,2H  /
;;
;;
;;      GO TO (1,2,3,4,5,6,7,8),ISTEP
;
(defpred (nextgoal STATE ?old STATE ?new)
    (nextgoal begin    calap)
    (nextgoal calap    xcapi)
    (nextgoal xcapi    ebsec)
    (nextgoal ebsec    regin)
    (nextgoal regin    rega)
    (nextgoal rega     xcapi1)
    (nextgoal xcapi1-y regas)
    (nextgoal xcapi1-n xcapi2)
    (nextgoal regas    xcapi2)
    (nextgoal xcapi2-y deta)
    (nextgoal xcapi2-n stop)
    (nextgoal deta     stop)
    )
;
(lisprule slut (goal stop) ->
  (err)
  )
;
(rule xcapi-rule
```

```
  (-> (goal xcapi) (done xcapi))
  )
;
(lisprule xcapi1-test (goal xcapi1) ->
  (display 'regional-analysis-sum-msg nil)
  (cond [(is-yes (dread)) (premiss '(done xcapi1-y))]
        [t (premiss '(done xcapi1-n))])
  )
;
(lisprule xcapi2-test (goal xcapi2) ->
  (display 'detail-analysis-msg nil)
  (cond [(is-yes (dread)) (premiss '(done xcapi2-y))]
        [t (premiss '(done xcapi2-n))])
  )
;
;(p xcapi-rule
;   (goal xcapi) -->
;     (make push ^cp xcapi ^ps ebsec regin rega xcapi-test-1)
;     )
;;
;; ----------------------------------------------------------------
;;  REGIONAL BACKGROUND PROGRAM SEGMENT
;; ----------------------------------------------------------------
;;
;;     1 ISTEP=ISTEP+1
;;       CALL EXEC(8,IEBSEC)
;;     2 ISTEP=ISTEP+1
;;
;; ----------------------------------------------------------------
;;  REGIONAL ANALYSIS PROGRAM SEGMENT
;; ----------------------------------------------------------------
;;
;;     3 ISTEP=ISTEP+1
;;       CALL EXEC(8,IREGA)
;;     4 ISTEP=ISTEP+1
;;
;; ----------------------------------------------------------------
;;  REGIONAL ANALYSIS SUMMARY SEGMENT
;; ----------------------------------------------------------------
;;
;(p xcapi-test-1-rule
;   (goal xcapi-test-1) -->
;     (call display regional-analysis-sum-msg)
;     (call yes-no xcapi)
;     )
;;
;(p xcapi-test-1-yin-rule
;   (goal xcapi-test-1)
;   (xcapi nil) -->
;     (make push ^cp xcapi-test-1 ^ps xcapi-test-2)
;     (remove 2)
;     )
;;
;(p xcapi-test-1-yang-rule
;   (goal xcapi-test-1)
;   (xcapi <n>) -->
;     (make push ^cp xcapi-test-1 ^ps regas xcapi-test-2)
;     (remove 2)
;     )
```

```
;;
;;       WRITE(LU(2),100)
(setq regional-analysis-sum-msg
      '(t t "Do you want to see the Regional Analysis Summary?" t t))
;;       READ(LU(1),*)IDO
;;       IF(IDO.NE.1)GO TO 50
;;     5 ISTEP=ISTEP+1
;;       CALL EXEC(8,IREGAS)
;;     6 ISTEP=ISTEP+1
;;
;; ------------------------------------------------------------------
;;  DETAILED ANALYSIS
;; ------------------------------------------------------------------
;;
;(p xcapi-test-2-rule
;   (goal xcapi-test-2) -->
;     (call display detail-analysis-msg)
;     (call yes-no xcapi)
;     )
;(p xcapi-test-2-yin-rule
;   (goal xcapi-test-2)
;   (xcapi nil) -->
;     (modify 1 ^1 done)
;     (remove 2)
;     )
;(p xcapi-test-2-yang-rule
;   (goal xcapi-test-2)
;   (xcapi t) -->
;     (remove 2)
;     (make push ^cp xcapi-test-2 ^ps deta)
;     )
;;
;;    50 WRITE(LU(2),105)
(setq detail-analysis-msg
      '(t t "Do you want to see the Detailed Analysis?" t t))
;;       READ(LU(1),*)JDO
;;       IF(JDO.NE.1)GO TO 60
;;       IF(IDO.NE.1)ISTEP=ISTEP+2
;;     7 ISTEP=ISTEP+1
;;       CALL EXEC(8,IDETA)
;;     8 ISTEP=ISTEP+1
;;
;;    60 CONTINUE
;;
;;       END
(terpr)
(princ "XCAPI.L Loaded")
(terpr)
```

EBSEC.L

```
;
;       ebsec.l     Regional Background Information Program
;                   Embayed Section of Atlantic Coastal Plain
;
;       Fortran System:   Robert Leighty    USAETL
;       OPS5 System:      Kenneth Hayes     Smart Systems Technology
;       DUCK System:      Kenneth Hayes     Smart Systems Technology
;
;;      PROGRAM EBSEC(5)
;;
;;  ********************************************************************
;;
;;   REGIONAL BACKGROUND INFORMATION FOR EMBAYED SECTION OF ATLANTIC
;;   COASTAL PLAIN.  SEGMENT OF PROGRAM CALAP.
;;
;;   RDL,  MAY 81
;;
;;  ********************************************************************
;;
;;       COMMON LU(5),MLAT1(3),MLON1(3),MLAT2(3),MLON2(3),ISTEP,MINR,
;;      *  MAXR,MINS,MAXS
;;
;;       COMMON  IRC,ISC,IRCE,ISCE,IRTR,ISTR,IRTRE,ISTRE,IRLC,
;;      *  IRLAC,ISLC,ISLTR,ISLAC,ISMC,ISMTR,IRTRR,IRTRM,IRTRS,ISTRR,
;;      *  ISTRM,ISTRS,ISMZ,IRT1O,IRT1C,IRT1TR,IST1O,IST1C,IST1TR,IRT2O,
;;      *  IST2O,IRT3O,IST3O,IRAC,IRACM,IRACML,IRACMH,IRARF,IRACD,IRACDH,
;;      *  IRS,IRSM,IRSMH,IRSD,IRSDH,IRSG,IRSGI,ISAC,ISACM,ISACMH,ISARF,
;;      *  ISACD,ISACDH,ISS,ISSM,ISSDH,ISSG,ISSGI,IRLTR,ISSD,ISSMH,
;;      *  IRGP(20),IRRQ(20),ISGP(20),ISRQ(20),IRTRI,ISTRI
;;
;;
;;       DIMENSION ICALAP(3)
;;
;;       DATA ICALAP/2HXC,2HAP,2HI /
;;
;;
;;  ------------------------------------------------------------------
;;   MENU PRESENTATION
;;  ------------------------------------------------------------------
;;
;;    20 WRITE(LU(2),22)
;
(lisprule ebsec-rule (goal ebsec) ->
  (for-first-ans (fetch '(study-region ?s ?p))
    (display 'ebsec-kickoff-msg (list ?s ?p))
    )
  (prog [a1 a2]
    top (for-first-ans (fetch '(study-region ?s ?p))
          (display 'ebsec-menu-msg (list ?s ?p))
          )
        (:= a1 (dread))
        (cond [(memq a1 '(1 p physiography phy pg))
                 (display 'ebsec-physiography-msg nil)]
              [(memq a1 '(2 lf landforms))
                 (display 'ebsec-landform-msg nil)]
              [(memq a1 '(3 g geology s soils geology/soils))
                 (display 'ebsec-geology-msg nil)]
              [(memq a1 '(4 c climate))
                 (display 'ebsec-climate-msg nil)]
```

```
                [(memq a1 '(5 d drainage))
                   (display 'ebsec-drainage-msg nil)]
                [(memq a1 '(6 v vegetation))
                   (display 'ebsec-vegetation-msg nil)]
                [(memq a1 '(7 l lu landuse))
                   (display 'ebsec-landuse-msg nil)]
                [(memq a1 '(8 h hydrology marine m marine/hydrology))
                   (display 'ebsec-hydrology-msg nil)]
                [(memq a1 '(9 e exit))
                   (return (premiss '(done stop)))]
                [(memq a1 '(0 ok)) (return)]
                [t (msg t "Illegal Answer, Please try again" t)
                   (go top)]
                )
    more (display 'ebsec-more-msg nil)
         (:= a2 (dread))
         (cond [(memq a2 '(1 detail more m d))
                   (display 'ebsec-physiography-more-msg nil)
                   (go top)]
               [(memq a2 '(2 another a)) (go top)]
               [(memq a2 '(3 continue c go g go-on))
                   (return)]
               [t (msg t "Sorry: " a2 " is not a legal response"
                         t "Please respond properly" t)
                   (go more)]
               )
         )
   (premiss '(done ebsec))
   )
;
(setq ebsec-kickoff-msg
      '(t t 10 "Your study area is in the " (o 1) " of" t
        5 "the " (o 2) " Physiographic Province. The" t
        5 "following menu will provide regional background" t
        5 "information on this physiographic section if you" t
        5 "desire." t t))
;;
;(p ebsec-menu-rule
;   (goal ebsec)
;   (study-region <s> <p>) -->
;     (call display ebsec-kickoff-msg <s> <p>)
;     (call display ebsec-menu-msg <s> <p>)
;     (make ebsec (accept))
;     (call clear)
;     )
;;
;;    25 WRITE(LU(2),26)
(setq ebsec-menu-msg
      '(t " ************************************************" t
         5 (o 1) ", " (o 2) t t
        10 "1 - Physiography" t
        10 "2 - Landforms" t
        10 "3 - Geology/Soils" t
        10 "4 - Climate" t
        10 "5 - Drainage" t
        10 "6 - Vegetation" t
        10 "7 - Landuse" t
        10 "8 - Marine Hydrology" t
        10 "9 - ****Terminate Program****" t
```

```
        10 "0 - Complete regional background phase" t t
        " **************************************************" t t
        5 "Please type your selection number" t t))
;;
;;      READ(LU(1),*)IPICK
;;      IF(IPICK.EQ.0.OR.IPICK.EQ.9)GO TO 1000
;;
;;      GO TO(110,120,130,140,150,160,170,180),IPICK
;;
;; ---------------------------------------------------------------
;;  PHYSIOGRAPHIC DESCRIPTION
;; ---------------------------------------------------------------
;;
;(p ebsec-physiography-rule
;   (goal ebsec)
;   (study-region |Embayed Section| |Atlantic Coastal Plain|)
;   (ebsec << 1 p physiography phy pg >>) -->
;     (call display ebsec-physiography-msg)
;     )
;;
;;  110 WRITE(LU(2),112)
(setq ebsec-physiography-msg
      '(t t 5 "PHYSIOGRAPHY OF THE EMBAYED SECTION," t
        5 "ATLANTIC COASTAL PLAIN PHYSIOGRAPHIC PROVINCE" t t
        10 "The Embayed Section of the Atlantic Coastal Plain" t
        10 "a submaturely dissected and partly submerged," t
        10 "terraced coastal plain." t t))
;;
;; ---------------------------------------------------------------
;;  ADDITIONAL INFORMATION?
;; ---------------------------------------------------------------
;;
;(p ebsec-more-rule
;   (goal ebsec)
;   (ebsec <n>) -->
;     (call display ebsec-more-msg)
;     (make ebsec-more (accept))
;     (call clear)
;     )
;;
;;      WRITE(LU(2),116)
(setq ebsec-more-msg
      '(t t 5 "Type:" t
        10 "1 - For additional information in this category" t
        10 "2 - For regional background information in a" t
        10 "      different category" t
        10 "3 - To complete your regional background" t
        10 "      information phase and proceed with regional" t
        10 "      analysis" t t))
;;
;;      READ(LU(1),*)IANSD
;;      GO TO(118,25,1000),IANSD
;;
;(p ebsec-more-ans-rule-1
;   (goal ebsec)
;   (ebsec)
;   (ebsec-more << 1 detail more m d >>) -->
;     (call display ebsec-physiography-more-msg)
;     (modify 3 ^2 2)
```

```
;       )
;(p ebsec-more-ans-rule-2
;    (goal ebsec)
;    (study-region <s> <p>)
;    (ebsec)
;    (ebsec-more << 2 another a >>) -->
;      (call display ebsec-menu-msg <s> <p>)
;      (modify 3 ^2 (accept))
;      (call clear)
;      (remove 4)
;      )
;(p ebsec-more-ans-rule-3
;    (goal ebsec)
;    (ebsec)
;    (ebsec-more << 3 continue c go g go-on >>) -->
;      (modify 1 ^1 done)
;      (remove 2 3)
;      )
;(p ebsec-more-ans-rule-4
;    (goal ebsec)
;    (ebsec-more <n>) -->
;      (write (crlf) "Sorry:" <n> "Not a legal response" (crlf)
;             "Please respond properly" (crlf))
;      (call display ebsec-more-msg)
;      (modify 2 ^2 (accept))
;      (call clear)
;      )
;;
;;   118 WRITE(LU(2),119)
(setq ebsec-physiography-more-msg
      '(t t 5 "SORRY NOTHING HERE YET. WILL ASSUME YOU WANT" t
        5 "MORE REGIONAL BACKGROUND INFORMATION" t t))
;;        GO TO 25
;;
;; ---------------------------------------------------------------
;;   LANDFORM DESCRIPTION
;; ---------------------------------------------------------------
;;
;(p ebsec-landform-rule
;    (goal ebsec)
;    (study-region |Embayed Section| |Atlantic Coastal Plain|)
;    (ebsec << 2 lf landforms >>) -->
;      (call display ebsec-landform-msg)
;      )
;;
;;   120 WRITE(LU(2),122)
(setq ebsec-landform-msg
      '(t t " MAJOR LANDFORMS OF THE EMBAYED SECTION," t
        " ATLANTIC COASTAL PLAIN:" t t
        5 "The major landform groups of the Embayed Section of the" t
        5 "Atlantic Coastal Plain are:" t
        10 "The coastal shoreline" t
        10 "Tidal rivers and associated features" t
        10 "Coastal Plain terraces" t
        10 "Recent alluvial deposits" t t
        5 "Associated with each of these major landform groups" t
        5 "individual landforms which will be described later." t))
;;
;;        WRITE(LU(2),116)
```

```
;;       READ(LU(1),*)IANSD
;;       GO TO(118,25,1000),IANSD
;;
;; ---------------------------------------------------------------
;;  GEOLOGY/SOILS DESCRIPTION
;; ---------------------------------------------------------------
;;
;(p ebsec-geology-rule
;   (goal ebsec)
;   (study-region !Embayed Section! !Atlantic Coastal Plain!)
;   (ebsec << 3 g geology s soils geology/soils >>) -->
;     (call display ebsec-geology-msg)
;     )
;;
;;  130 WRITE(LU(2),132)
(setq ebsec-geology-msg
      '(t t " GEOLOGY AND SOILS OF THE EMBAYED SECTION," t
        " ATLANTIC COASTAL PLAIN" t t
        5 "GEOLOGY" t
        5 "  The bedrock underlying the Coastal Plain sediments" t
        5 " consists of granite/gneiss, gneiss and schist," t
        5 " generally easterly tilted, depending on the landform." t
        5 " These formations are covered with marine or fluvial" t
        5 " sediments." t t
        5 "SOILS:" t
        5 " The soils of the Embayed Section of the Coastal Plain" t
        5 " consist of layers of marl (lime), silt and clay, and" t
        5 " sand-clay and gravels in the higher elevations." t t))
;;
;;       WRITE(LU(2),116)
;;       READ(LU(1),*)IANSD
;;       GO TO (118,25,1000),IANSD
;;
;; ---------------------------------------------------------------
;;  CLIMATE DESCRIPTION
;; ---------------------------------------------------------------
;;
;(p ebsec-climate-rule
;   (goal ebsec)
;   (study-region !Embayed Section! !Atlantic Coastal Plain!)
;   (ebsec << 4 c climate >>) -->
;     (call display ebsec-climate-msg)
;     )
;;
;;  140 WRITE(LU(2),142)
(setq ebsec-climate-msg
      '(t t " CLIMATE OF THE EMBAYED SECTION," t
        " ATLANTIC COASTAL PLAIN" t t
        10 "The climate in the Embayed Section of the Atlantic" t
        10 "Coastal Plain is humid temperate. Temperatures vary" t
        10 "seasonally with a difference of 36 degrees mean" t
        10 "winter to summer temperature." t t))
;;
;;       WRITE(LU(2),116)
;;       READ(LU(1),*)IANSD
;;       GO TO (118,25,1000),IANSD
;;
;; ---------------------------------------------------------------
;;  DRAINAGE DESCRIPTION
```

```
;; ------------------------------------------------------------
;;
;(p ebsec-drainage-rule
;   (goal ebsec)
;   (study-region |Embayed Section| |Atlantic Coastal Plain|)
;   (ebsec << 5 d drainage >>) -->
;     (call display ebsec-drainage-msg)
;     )
;;
;;  150 WRITE(LU(2),152)
(setq ebsec-drainage-msg
      '(t t " DRAINAGE OF THE EMBAYED SECTION," t
        " ATLANTIC COASTAL PLAIN" t t
        5 " The drainage patterns of the Embayed Section of the" t
        5 "Atlantic Coastal Plain are defined by the major" t
        5 "landform groups:" t t
        8 "Tidal flats, marshes, and meander systems," t
        8 "Lowlands and low terraces without developed patterns" t
        8 "and Upper terraces with internal to coarse dendritic" t
        8 "  patterns and some small poorly drained sections." t t))
;;
;;      WRITE(LU(2),116)
;;      READ(LU(1),*)IANSD
;;      GO TO (118,25,1000),IANSD
;;
;; ------------------------------------------------------------
;;  VEGETATION DESCRIPTION
;; ------------------------------------------------------------
;;
;(p ebsec-vegetation-rule
;   (goal ebsec)
;   (study-region |Embayed Section| |Atlantic Coastal Plain|)
;   (ebsec << 6 v vegetation >>) -->
;     (call display ebsec-vegetation-msg)
;     )
;;
;;  160 WRITE(LU(2),162)
(setq ebsec-vegetation-msg
      '(t t " VEGETATION OF THE EMBAYED SECTION," t
        " ATLANTIC COASTAL PLAIN" t t
        5 " The natural vegetation of the Embayed Section of the" t
        5 "Atlantic Coastal Plain is ill-defined because of" t
        5 "cultural activity. Forest stands are restricted mainly" t
        5 "to preserves, parks, and areas of poor drainage or" t
        5 "steep slopes such as found in gullies or terrace faces." t
        5 "Grasses, etc. are found in cleared areas, while marsh-" t
        5 "type vegetation occur in tidal areas." t t))
;;      WRITE(LU(2),116)
;;      READ(LU(1),*)IANSD
;;      GO TO (118,25,1000),IANSD
;;
;; ------------------------------------------------------------
;;  LANDUSE DESCRIPTION
;; ------------------------------------------------------------
;;
;(p ebsec-landuse-rule
;   (goal ebsec)
;   (study-region |Embayed Section| |Atlantic Coastal Plain|)
;   (ebsec << 7 1 lu landuse >>) -->
```

```
;       (call display ebsec-landuse-msg)
;       )
;;
;;   170 WRITE(LU(2),172)
(setq ebsec-landuse-msg
       '(t t " LANDUSE OF THE EMBAYED SECTION," t
         " ATLANTIC COASTAL PLAIN" t t
         5 " The landuse of the Embayed Section of the Atlantic" t
         5 "Coastal Plain is predominantly agricultural or" t
         5 "urban/suburban. Heavy industry is found mainly in" t
         5 "urban centers near the coast or major rivers/bays." t
         5 "Dense transportation networks connect major urban" t
         5 "centers." t t))
;;       WRITE(LU(2),116)
;;       READ(LU(1),*)IANSD
;;       GO TO (118,25,1000),IANSD
;;
;; ---------------------------------------------------------------
;;  MARINE HYDROLOGY DESCRIPTION
;; ---------------------------------------------------------------
;;
;(p ebsec-hydrology-rule
;    (goal ebsec)
;    (study-region !Embayed Section! !Atlantic Coastal Plain!)
;    (ebsec << 8 h hydrology marine m marine/hydrology >>) -->
;      (call display ebsec-hydrology-msg)
;      )
;;
;;   180 WRITE(LU(2),182)
(setq ebsec-hydrology-msg
       '(t t " MARINE HYDROLOGY OF THE EMBAYED SECTION," t
         " ATLANTIC COASTAL PLAIN" t t
         10 "The marine hydrology of the Embayed Section of the" t
         5 "Atlantic Coastal Plain will be added later to describe" t
         5 "marine conditions along the coastline and tidal rivers." t t))
;;       WRITE(LU(2),116)
;;       READ(LU(1),*)IANSD
;;       GO TO (118,25,1000),IANSD
;;
;; ---------------------------------------------------------------
;;  RETURN TO EXECUTIVE PROGRAM
;; ---------------------------------------------------------------
;;
;(p ebsec-done-rule
;    (goal ebsec)
;    (ebsec << 0 ok >>) -->
;      (modify 1 ^1 done)
;      (remove 2)
;      )
;;
;(p ebsec-quit-rule
;    (goal ebsec)
;    (ebsec << 9 e exit >>) -->
;      (remove 1 2)
;      (halt)
;      )
;;
;;  1000 IF(IPICK.EQ.9)GO TO 1005
;;       CALL EXEC(8,ICALAP)
```

```
;;
;; 1005 END
(terpr)
(princ "EBSEC.L Loaded")
(terpr)
```

REGA.L

```
;
;       rega.l      Regional Analysis Program
;
;       Fortran System:   Robert Leighty    USAETL
;       OPS5 System:      Kenneth Hayes     Smart Systems Technology
;       DUCK System:      Kenneth Hayes     Smart Systems Technology
;
;;      PROGRAM REGA(5)
;;
;; ***************************************************************************
;;
;;  REGIONAL ANALYSIS SEGMENT FOR PROGRAM CALAP.
;;
;;  RDL, MAY 81
;;
;; ***************************************************************************
;;
;;        ADMIN NOTES:
;;          100 STMTS - INTRO & <20'
;;          200 STMTS - LOW TERRACE
;;          300 STMTS - MID TERRACE
;;          400 STMTS - HIGH TERRACE
;;          500 STMTS - DRAINAGE
;;          700 STMTS - SPECIAL
;;         1000 STMTS - SUMMARY
;;
;;       COMMON LU(5),MLAT1(3),MLON1(3),MLAT2(3),MLON2(3),ISTEP,MINR,
;;      *  MAXR,MINS,MAXS
;;
;;       COMMON  IRC,ISC,IRCE,ISCE,IRTR,ISTR,IRTRE,ISTRE,IRLC,
;;      *  IRLAC,ISLC,ISLTR,ISLAC,ISMC,ISMTR,IRTRR,IRTRM,IRTRS,ISTRR,
;;      *  ISTRM,ISTRS,ISMZ,IRT1O,IRT1C,IRT1TR,IST1O,IST1C,IST1TR,IRT2O,
;;      *  IST2O,IRT3O,IST3O,IRAC,IRACM,IRACML,IRACMH,IRARF,IRACD,IRACDH,
;;      *  IRS,IRSM,IRSMH,IRSD,IRSDH,IRSG,IRSGI,ISAC,ISACM,ISACMH,ISARF,
;;      *  ISACD,ISACDH,ISS,ISSM,ISSDH,ISSG,ISSGI,IRLTR,ISSD,ISSMH,
;;      *  IRGP(20),IRRG(20),ISGP(20),ISRG(20),IRTRI,ISTRI
;;
;;       DIMENSION ICALAP(3),IYES(2),INO(2),IBEGIN(3)
;;
;;       DATA ICALAP/2HXC,2HAP,2HI /
;;       DATA IYES/2HYE,2HS /,INO/2HNO,2H  /
;;       DATA IBEGIN/2HBE,2HGI,2HN /
;;
;; ---------------------------------------------------------------------
;;  INITIALIZE DECISION VARIABLES
;; ---------------------------------------------------------------------
;;
;;       CALL REGIN
;;
;; ---------------------------------------------------------------------
;;  INTRODUCTION
;; ---------------------------------------------------------------------
;;
;
(lisprule rega-rule (goal rega) ->
  (display 'rega-main-msg nil)
  (pause)
  (display 'rega-sr-elv-msg nil)
  (pause)
```

```
  (prog [a1 a2 a3 a4]
   rtop (display 'rega-sr-elv-msg-2 nil)
        (msg t "min:")
        (:= a1 (dread))
        (msg t "max:")
        (:= a2 (dread))
        (cond [(and (numberp a1) (numberp a2) (lessp a1 a2))
                 (display 'rega-sr-elv-prt-msg `(,a1 ,a2))
                 (cond [(is-yes (dread))
                          (go atop)]
                       [t (display 'rega-sr-elv-again-msg nil)])]
              [t (msg t "Bad elevation values" t "Try again" t)])
        (go rtop)
   atop (display 'rega-sa-elv-msg nil)
        (pause)
        (display 'rega-sa-elv-msg-2 nil)
        (msg t "min:")
        (:= a3 (dread))
        (msg t "max:")
        (:= a4 (dread))
        (cond [(and (numberp a3) (numberp a4)
                    (lessp a3 a4)
                    (=< a1 a3) (=< a4 a2))
                 (display 'rega-sr-elv-prt-msg `(,a3 ,a4))
                 (cond [(is-yes (dread))
                          (premiss `(answer srmin ,a1))
                          (premiss `(answer srmax ,a2))
                          (premiss `(answer samin ,a3))
                          (premiss `(answer samax ,a4))
                          (return)]
                       [t (display 'rega-sr-elv-again-msg nil)
                          (go atop)])]
              [t (display 'rega-sa-elv-kvetch-msg `(,a1 ,a2 ,a3 ,a4))
                 (go rtop)])
        )
  (display 'rega-map-photo-msg nil)
  (pause)
  (cond [(lessp (gans srmin) 20) (premiss '(goal ocean-tidal-river))])
  (cond [(and (lessp (gans srmin) 100) (greaterp (gans srmax) 20))
         (premiss '(goal low-ter))])
  (cond [(and (lessp (gans srmin) 170) (greaterp (gans srmax) 100))
         (premiss '(goal mid-ter))])
  (cond [(and (lessp (gans srmin) 270) (greaterp (gans srmax) 170))
         (premiss '(goal high-ter))])
  (premiss '(goal drainage))
  (premiss '(goal special))
  (premiss '(goal listing))
  (premiss '(done rega))
  )
;
(lisprule rega-o-t-r (goal ocean-tidal-river) ->
  (premiss '(ask-n s-r-coastal-shoreline
                   rega-shore-ques-1
                   s-a-coastal-shoreline))
  (cond [(and (gans s-r-coastal-shoreline)
              (lessp (gans samin) 20))
         (user 's-a-coastal-shoreline 'rega-shore-ques-2)
         (cond [(gans s-a-coastal-shoreline)
                  (user 's-a-coastal-shoreline-ext
```

```
                        'rega-sa-shore-ext-ques)]
                [t (user 's-r-coastal-shoreline-ext
                        'rega-sr-shore-ext-ques)])
          ])
  (premiss '(ask-n s-r-tidal-river rega-tidal-river-q-1 s-a-tidal-river))
  (cond [(gans s-r-tidal-river) (premiss '(goal triver))]
        [t (premiss '(goal ntriver))])
  )
;
(lisprule rega-notriver (goal ntriver) ->
  (display 'rega-no-c-no-tr-low-msg-1 nil)
  (premiss '(ask-n s-r-low-coastal-shoreline
                   rega-no-c-no-tr-low-msg-2
                   s-a-low-coastal-shoreline))
  (cond [(gans s-r-low-coastal-shoreline)]
        [(progn (premiss '(ask-n s-r-low-tidal-river
                                 rega-no-c-no-tr-low-msg-3
                                 s-a-low-tidal-river))
                (gans s-r-low-tidal-river))]
        [(progn (premiss '(ask-n s-r-low-alluvial-channel
                                 rega-no-c-no-tr-low-msg-4
                                 s-a-low-alluvial-channel))
                (gans s-r-low-alluvial-channel))]
        [t (display 'rega-no-c-no-tr-low-msg-5 nil)
           (display 'rega-no-c-no-tr-low-msg-6 nil)])
  (cond [(lessp (gans samin) 20)
           (display 'rega-120-msg-1 nil)
           (cond [(gans s-r-low-coastline)
                    (user 's-a-low-coastal-shoreline
                          'rega-120-msg-2)])
           (cond [(gans s-r-low-tidal-river)
                    (user 's-a-low-tidal-river
                          'rega-120-msg-3)])
           (cond [(gans s-r-low-alluvial-channel)
                    (user 's-a-low-alluvial-channel
                          'rega-120-msg-4)])
           (premiss '(goal low-sa-max))]
        [t (premiss '(goal low-sr-max))])
  )
;
(lisprule rega-triver (goal triver) ->
  (premiss '(ask-n s-r-tidal-river-islands
                   rega-sr-tr-island-q
                   s-a-tidal-river-islands))
  (premiss '(ask-n s-r-tidal-river-rock
                   rega-sr-tr-rock-q
                   s-a-tidal-river-rock))
  (premiss '(ask-n s-r-tidal-river-mud
                   rega-sr-tr-mud-q
                   s-a-tidal-river-mud))
  (premiss '(ask-n s-r-tidal-river-sand
                   rega-sr-tr-sand-q
                   s-a-tidal-river-sand))
  (cond [(lessp (gans srmin) 20)
           (user 's-a-tidal-river 'rega-sa-tidal-river-q)
           (cond [(gans s-a-tidal-river)
                    (user 's-a-tidal-river-islands
                          'rega-sa-tidal-river-islands-q)
                    (user 's-a-tidal-river-rock
```

```
                              'rega-sa-tidal-river-rock-q)
                        (user 's-a-tidal-river-mud
                              'rega-sa-tidal-river-mud-q)
                        (user 's-a-tidal-river-sand
                              'rega-sa-tidal-river-sand-q)
                        (user 's-a-tidal-river-ext
                              'rega-sr-tr-sa-tr-msg)]
                    [t (user 's-r-tidal-river-ext
                              'rega-sr-tr-no-sa-tr-msg)])
              ])
    (premiss '(goal low-sa-max))
    )
;
(lisprule rega-low-sa-max (goal low-sa-max) ->
    (cond [(not (lessp (gans samax) 20))]
          [(user 's-a-max-coastal-shoreline
                 'rega-1120-msg-1)]
          [(user 's-a-max-tidal-river
                 'rega-1120-msg-2)]
          [(user 's-a-max-un-associated
                 'rega-1120-msg-3)])
    (premiss '(goal low-sr-max))
    )
;
(lisprule rega-low-sr-max (goal low-sr-max) ->
    (msg t "No statements for low study region maximum elevation" t)
    )
;
(lisprule rega-low-ter (goal low-ter) ->
    (display 'rega-sr-ltr-msg-1 nil)
    (display 'rega-sr-ltr-msg-2 nil)
    (pause)
    (display 'rega-sr-ltr-msg-3 nil)
    (premiss '(ask-n s-r-low-terraces
                     rega-sr-ltr-msg-4
                     s-a-low-terraces))
    (cond [(gans s-r-low-terraces)
           (cond [(gans s-r-coastal-shoreline)
                  (premiss '(ask-n s-r-low-terraces-near-coast
                                   rega-sr-ltr-msg-5
                                   s-a-low-terraces-near-coast))])
           (cond [(gans s-r-tidal-river)
                  (premiss '(ask-n s-r-low-terraces-near-tidal-river
                                   rega-sr-ltr-msg-6
                                   s-a-low-terraces-near-tidal-river))])
           ])
    (cond [(and (lessp (gans samin) 100) (greaterp (gans samax) 20))
           (premiss '(ask s-a-low-terraces rega-sa-ltr-msg-1 nil))
           (cond [(gans s-a-coastal-shoreline)
                  (premiss '(ask s-a-low-terraces-near-coast
                                 rega-sa-ltr-msg-2 nil))])
           (cond [(gans s-a-tidal-river)
                  (premiss '(ask s-a-low-terraces-near-tidal-river
                                 rega-sa-ltr-msg-3 nil))])
          ])
    )
;
(lisprule rega-mid-ter (goal mid-ter) ->
    (display 'rega-sr-mtr-msg-1 nil)
```

```
  (display 'rega-sr-mtr-msg-2 nil)
  (pause)
  (display 'rega-sr-mtr-msg-3 nil)
  (premiss '(ask-n s-r-mid-terraces
                   rega-sr-mtr-msg-4
                   s-a-mid-terraces))
  (cond [(and (lessp (gans samin) 170) (greaterp (gans samax) 100))
         (premiss '(ask s-a-mid-terraces rega-sa-mtr-msg-1 nil))])
  )
;
(lisprule rega-high-ter (goal high-ter) ->
  (display 'rega-sr-htr-msg-1 nil)
  (display 'rega-sr-htr-msg-2 nil)
  (pause)
  (display 'rega-sr-htr-msg-3 nil)
  (premiss '(ask-n s-r-high-terraces
                   rega-sr-htr-msg-4
                   s-a-high-terraces))
  (cond [(and (lessp (gans samin) 270) (greaterp (gans samax) 170))
         (premiss '(ask s-a-high-terraces rega-sa-htr-msg-1 nil))])
  )
;
(lisprule rega-drainage (goal drainage) ->
  (display 'rega-sr-drg-msg-1 nil)
  (pause)
  (cond [(user 's-r-rivers 'rega-sr-drg-msg-2)
         (cond [(user 's-r-meandering-rivers 'rega-sr-drg-msg-3)
                (cond [(greaterp (gans srmax) 100)
                       (user 's-r-low-meandering-rivers
                             'rega-sr-drg-msg-4)])
                (cond [(greaterp (gans srmin) 100)
                       (user 's-r-high-meandering-rivers
                             'rega-sr-drg-msg-5)])])
         (user 's-r-rapid/fall-rivers 'rega-sr-drg-msg-6)
         (cond [(user 's-r-deflected-rivers 'rega-sr-drg-msg-7)
                (cond [(greaterp (gans srmin) 100)
                       (user 's-r-high-deflected-rivers
                             'rega-sr-drg-msg-8)])])]
        [t (premiss '(answer s-a-rivers nil))])
  (cond [(user 's-r-streams 'rega-sr-drg-msg-9)
         (cond [(user 's-r-meandering-streams 'rega-sr-drg-msg-10)
                (cond [(greaterp (gans srmax) 100)
                       (user 's-r-high-meandering-streams
                             'rega-sr-drg-msg-11)])])
         (cond [(user 's-r-distributary-segments
                      'rega-sr-drg-msg-12)
                (cond [(greaterp (gans srmax) 100)
                       (user 's-r-high-distributary-segments
                             'rega-sr-drg-msg-13)])])
         (cond [(user 's-r-strange-gradients 'rega-sr-drg-msg-14)
                (user 's-r-strange-gradients-incised
                      'rega-sr-drg-msg-15)])]
        [t (premiss '(answer s-a-streams nil))])
  (cond [(and (gans s-r-rivers)
              (user 's-a-rivers 'rega-sa-drg-msg-1))
         (cond [(user 's-a-meandering-rivers 'rega-sa-drg-msg-2)
                (cond [(greaterp (gans samax) 100)
                       (user 's-a-high-meandering-rivers
                             'rega-sa-drg-msg-3)])])
```

```
            (user 's-a-rapid/fall-rivers 'rega-sa-drg-msg-4)
            (cond [(user 's-a-deflected-rivers 'rega-sa-drg-msg-5)
                   (cond [(greaterp (gans samax) 100)
                          (user 's-a-high-deflected-rivers
                                'rega-sa-drg-msg-6)])])]
  (cond [(and (gans s-r-streams)
              (user 's-a-streams 'rega-sa-drg-msg-7))
            (cond [(user 's-a-meandering-streams
                         'rega-sa-drg-msg-8)
                   (cond [(greaterp (gans samax) 100)
                          (user 's-a-high-meandering-streams
                                'rega-sa-drg-msg-9)])]
            (cond [(user 's-a-distributary-segments
                         'rega-sa-drg-msg-10)
                   (cond [(greaterp (gans samax) 100)
                          (user 's-a-hig' -distributary-segments
                                'rega-sa  rg-msg-11)])])
            (cond [(user 's-a-strange-gradients
                         'rega-sa-drg-msg-12)
                   (user 's-a-strange-gradients-incised
                         'rega-sa-drg-msg-13)])])
  )
;
(lisprule rega-special (goal special) ->
  (prog [a1]
        (display 'rega-special-msg nil)
        (display 'rega-spc-pits-msg-1 nil)
        (pause)
        (display 'rega-spc-pits-msg-2 nil)
        (:= a1 (dread))
        (premiss `(answer s-r-gravel-pits ,(and a1 t)))
        (premiss `(spanswer s-r-gravel-pits (tup .,a1)))
        (display 'rega-spc-pits-msg-3 nil)
        (:= a1 (dread))
        (premiss `(answer s-a-gravel-pits ,(and a1 t)))
        (premiss `(spanswer s-a-gravel-pits (tup .,a1)))
        (display 'rega-spc-quarry-msg-1 nil)
        (:= a1 (dread))
        (premiss `(answer s-r-quarry ,(and a1 t)))
        (premiss `(spanswer s-r-quarry (tup .,a1)))
        (display 'rega-spc-quarry-msg-2 nil)
        (:= a1 (dread))
        (premiss `(answer s-a-quarry ,(and a1 t)))
        (premiss `(spanswer s-a-quarry (tup .,a1)))
        )
  )
;
(lisprule rega-listing (goal listing) ->
  (display 'rega-listing-start-msg nil)
  (pause)
  (display 'rega-list-header-msg nil)
  (display 'rega-list-elv-msg
           (list (gans srmin) (gans srmax) (gans samin) (gans samax)))
  (display 'rega-list-sr-msg-1 nil)
  (display-tf 'rega-list-sr-msg-2
              (list (gans+ s-r-coastal-shoreline)
                    (gans+ s-r-coastal-shoreline-ext)
                    (gans+ s-r-tidal-river)
                    (gans+ s-r-tidal-river-ext)))
```

```
(display-tf 'rega-list-sr-msg-3
            (list (gans+ s-r-tidal-river-islands)
                  (gans+ s-r-low-coastal-shoreline)
                  (gans+ s-r-low-tidal-river)
                  (gans+ s-r-low-alluvial-channel)))
(display-tf 'rega-list-sr-msg-4
            (list (gans+ s-r-tidal-river-rock)
                  (gans+ s-r-tidal-river-mud)
                  (gans+ s-r-tidal-river-sand)
                  (gans+ s-r-low-terraces)))
(display-tf 'rega-list-sr-msg-5
            (list (gans+ s-r-low-terraces-near-coast)
                  (gans+ s-r-low-terraces-near-tidal-river)
                  (gans+ s-r-mid-terraces)
                  (gans+ s-r-high-terraces)))
(display-tf 'rega-list-sr-msg-6
            (list (gans+ s-r-rivers)
                  (gans+ s-r-meandering-rivers)
                  (gans+ s-r-low-meandering-rivers)))
(display-tf 'rega-list-sr-msg-7
            (list (gans+ s-r-high-meandering-rivers)
                  (gans+ s-r-rapid/fall-rivers)
                  (gans+ s-r-deflected-rivers)))
(display-tf 'rega-list-sr-msg-8
            (list (gans+ s-r-high-deflected-rivers)
                  (gans+ s-r-streams)
                  (gans+ s-r-meandering-streams)
                  (gans+ s-r-high-meandering-streams)))
(display-tf 'rega-list-sr-msg-9
            (list (gans+ s-r-distributary-segments)
                  (gans+ s-r-high-distributary-segments)
                  (gans+ s-r-strange-gradients)
                  (gans+ s-r-strange-gradients-incised)))
(display-and 'rega-list-sr-msg-10
             (gans s-r-gravel-pits)
             (gansp s-r-gravel-pits))
(display-and 'rega-list-sr-msg-11
             (gans s-r-quarry)
             (gansp s-r-quarry))
(display 'rega-list-sa-msg-1 nil)
(display-tf 'rega-list-sa-msg-2
            (list (gans+ s-a-coastal-shoreline)
                  (gans+ s-a-coastal-shoreline-ext)
                  (gans+ s-a-tidal-river)
                  (gans+ s-a-tidal-river-ext)))
(display-tf 'rega-list-sa-msg-3
            (list (gans+ s-a-tidal-river-islands)
                  (gans+ s-a-low-coastal-shoreline)
                  (gans+ s-a-low-tidal-river)
                  (gans+ s-a-low-alluvial-channel)))
(display-tf 'rega-list-sa-msg-4
            (list (gans+ s-a-max-coastal-shoreline)
                  (gans+ s-a-max-tidal-river)
                  (gans+ s-a-max-un-associated)
                  (gans+ s-a-tidal-river-rock)))
(display-tf 'rega-list-sa-msg-5
            (list (gans+ s-a-tidal-river-mud)
                  (gans+ s-a-tidal-river-sand)
                  (gans+ s-a-low-terraces)
```

```
                    (gans+ s-a-low-terraces-near-coast)))
  (display-tf 'rega-list-sa-msg-6
              (list (gans+ s-a-low-terraces-near-tidal-river)
                    (gans+ s-a-mid-terraces)
                    (gans+ s-a-high-terraces)
                    (gans+ s-a-rivers)))
  (display-tf 'rega-list-sa-msg-7
              (list (gans+ s-a-meandering-rivers)
                    (gans+ s-a-high-meandering-rivers)
                    (gans+ s-a-rapid/fall-rivers)
                    (gans+ s-a-deflected-rivers)))
  (display-tf 'rega-list-sa-msg-8
              (list (gans+ s-a-high-deflected-rivers)
                    (gans+ s-a-streams)
                    (gans+ s-a-meandering-streams)
                    (gans+ s-a-high-meandering-streams)))
  (display-tf 'rega-list-sa-msg-9
              (list (gans+ s-a-distributary-segments)
                    (gans+ s-a-high-distributary-segments)
                    (gans+ s-a-strange-gradients)
                    (gans+ s-a-strange-gradients-incised)))
  (display-and 'rega-list-sa-msg-10
               (gans s-a-gravel-pits)
               (gansp s-a-gravel-pits))
  (display-and 'rega-list-sa-msg-11
               (gans s-a-quarry)
               (gansp s-a-quarry))
  )
;
;(p rega-rule-1
;   (goal rega) -->
;     (call display rega-main-msg)
;     (make rega sr-elv)
;     (call pause)
;     )
;(p rega-done-rule
;   (goal rega)
;   (rega done) -->
;     (modify 1 ^1 done)
;     (remove 2)
;     )
;;
;;      WRITE(LU(2),5)
(setq rega-main-msg
      '(t t " ************REGIONAL ANALYSIS*******************" t t
        5 "We are about to begin the regional analysis phase of" t
        5 "landform analysis. In this phase of the investigation" t
        5 "we will define which of the major landform groups are" t
        5 "found in our 'study region' (the area covered by the" t
        5 "map sheet) and our 'study area'. In the detailed" t
        5 "analysis phase following this regional analysis phase" t
        5 "we will identify and partition the individual terrain" t
        5 "or topographic forms of the landform groups within the" t
        5 "study area." t t))
;;
;;    6 WRITE(LU(2),7)
;;(setq rega-msg
;;      '(t t " Type 'go' to continue" t t t t))
;;      PAUSE 1
```

```
;;
;; ------------------------------------------------------------
;;  ENTER REGIONAL MINIMUM AND MAXIMUM TERRAIN ELEVATIONS
;; ------------------------------------------------------------
;;
;(p sr-elv-rule-1
;    (goal rega)
;    (rega sr-elv) -->
;      (call display rega-sr-elv-msg)
;      (call pause)
;      (call display rega-sr-elv-msg-2)
;      (make sr-elv)
;      )
;;
;;     9 WRITE(LU(2),10)
(setq rega-sr-elv-msg
      '(t t t "The information we need to begin the regional" t
        " analysis relates to terrain elevations on your map" t
        " sheet." t t
        5 "Locate the minimum and maximum terrain elevations" t
        5 "on the map sheet. In general, the study region minimum" t
        5 "elevation will be found on the edge of the map sheet." t t))
;;       WRITE(LU(2),7)
;;       PAUSE 2
;;
;(p sr-elv-rule-2
;    (goal rega)
;    (rega sr-elv)
;    (sr-elv) -->
;      (write (crlf) "min max:")
;      (make s-r-elevation (accept) (accept))
;      (call clear)
;      )
;;
;;       WRITE(LU(2),15)
(setq rega-sr-elv-msg-2
      '(t t " Now enter the minimum and maximum study region elevation." t
        " Hereafter these elevations will" t
        " be referred to as the study region minimum and maximum." t t))
;;    20 READ(LU(1),*)MINR,MAXR
;;       WRITE(LU(2),22)MINR,MAXR
;;
;(p sr-elv-rule-3
;    (goal rega)
;    (rega sr-elv)
;    (sr-elv)
;    (s-r-elevation {<=> 1 <min>} {<=> 1 >= <min> <max>}) -->
;      (call display rega-sr-elv-prt-msg <min> <max>)
;      (call yes-no sr-elv-check)
;      )
;;
(setq rega-sr-elv-prt-msg
      '(t t " The minimum and maximum values typed were:" t t
        5 (o 1) " and " (o 2) t t
        " Are these values correct?" t t))
;;
;(p sr-elv-rule-4
;    (goal rega)
;    (rega sr-elv)
```

```
;    (sr-elv)
;    (s-r-elevation) -->
;      (write (crlf) "Bad elevation values" (crlf) "Try again" (crlf))
;      (modify 3 ^1 sr-elv)
;      (remove 4)
;      )
;;
;;       READ(LU(1),24)IANSD
;;    24 FORMAT(2A2)
;;       IF(IANSD.EQ.IYES)GO TO 30
;;       WRITE(LU(2),26)
;;
;(p sr-elv-ok-rule
;    (goal rega)
;    (rega sr-elv)
;    (sr-elv)
;    (s-r-elevation)
;    (sr-elv-check t) -->
;      (modify 2 ^2 sa-elv)
;      (remove 3 5)
;      )
;;
(setq rega-sr-elv-again-msg
      '(" OK, Retype the correct elevation values." t t))
;;       GO TO 20
;;
;(p sr-elv-again-rule
;    (goal rega)
;    (rega sr-elv)
;    (sr-elv)
;    (s-r-elevation)
;    (sr-elv-check) -->
;      (call display rega-sr-elv-again-msg)
;      (modify 3 ^1 sr-elv)
;      (remove 4 5)
;      )
;;
;;  ------------------------------------------------------------
;;   ENTER STUDY AREA MINIMUM AND MAXIMUM TERRAIN ELEVATIONS
;;  ------------------------------------------------------------
;;
;(p sa-elv-rule-1
;    (goal rega)
;    (rega sa-elv) -->
;      (call display rega-sa-elv-msg)
;      (call pause)
;      (call display rega-sa-elv-msg-2)
;      (make sa-elv)
;      )
;;
;;    30 WRITE(LU(2),35)
(setq rega-sa-elv-msg
      '(t t " The next information we need is also related to" t
        " terrain elevations:" t t
        5 "Locate the minimum and maximum terrain elevations" t
        5 "for the 'study area' from the map sheet. Note that" t
        5 "the minimum elevation will usually be found on the" t
        5 "edge of the study area." t t))
;;       WRITE(LU(2),7)
```

```
;;       PAUSE 3
;;
;(p sa-elv-rule-2
;    (goal rega)
;    (rega sa-elv)
;    (sa-elv) -->
;      (write (crlf) "min max:")
;      (make s-a-elevation (accept) (accept))
;      (call clear)
;      )
;;
;;       WRITE(LU(2),37)
(setq rega-sa-elv-msg-2
      '(t t " Now enter these elevation values into the system." t t))
;;
;(p sa-elv-rule-3
;    (goal rega)
;    (rega sa-elv)
;    (sa-elv)
;    (s-a-elevation {<= 1 <min>} {<= 1 >= <min> <max>})
;    (s-r-elevation {<= <min> <rmin>} {>= <max> <rmax>}) -->
;      (call display rega-sr-elv-prt-msg <min> <max>)
;      (call yes-no sa-elv-check)
;      )
;;
;;    40 READ(LU(1),*)MINS,MAXS
;;   CHECK ELEVATION VALUES FOR CONSISTENCY
;;       IF((MINR.LE.MINS).AND.(MAXR.GE.MAXS))GO TO 46
;;   IF AN INCONSISTENCY IS DETECTED, THEN SHOW ERROR AND REPEAT INPUT
;;       IF(MINR.GT.MINS)WRITE(LU(2),42)MINR,MINS
;;       IF(MAXR.LT.MAXS)WRITE(LU(2),44)MAXR,MAXS
;;
;(p sa-elv-rule-4
;    (goal rega)
;    (rega sa-elv)
;    (sa-elv)
;    (s-a-elevation <samin> <samax>)
;    (s-r-elevation <srmin> <srmax>) -->
;      (call display rega-sa-elv-kvetch-msg <srmin> <samin> <srmax> <samax>)
;      (modify 2 ^1 sr-elv)
;      (remove 3 4 5)
;      )
;
(setq rega-sa-elv-kvetch-msg
      '(t t " STUDY REGION MINIMUM ELEVATION MUST BE EQUAL" t
        " TO OR LESS THAN STUDY AREA MINIMUM ELEVATION." t t
        " YOUR VALUES WERE:" t
        5 "Study region minimum = " (o 1) t
        5 "Study area minimum   = " (o 2) t t
        t t " STUDY REGION MAXIMUM ELEVATION MUST BE GREATER" t
        " THAN OR EQUAL TO THE STUDY AREA MAXIMUM ELEVATION." t t
        " YOUR VALUES WERE:" t
        5 "Study region maximum = " (o 3) t
        5 "Study area maximum   = " (o 4) t t
        " LET'S TRY AGAIN FROM THE TOP" t t))
;;
;(p sa-elv-ok-rule
;    (goal rega)
;    (rega sa-elv)
```

```
;    (sa-elv)
;    (s-a-elevation)
;    (sa-elv-check t) -->
;      (modify 2 ^2 map-photo)
;      (remove 3 5)
;      )
;;
;;       GO TO 9
;;   PRINT OF CONSISTENT ELEVATION VALUES & CHECK FOR CORRECTIONS
;;    46 WRITE(LU(2),22)MINS,MAXS
;;       READ(LU(1),24)IANSD
;;       IF(IANSD.EQ.IYES)GO TO 50
;;       WRITE(LU(2),26)
;;       GO TO 40
;;
;(p sa-elv-again-rule
;    (goal rega)
;    (rega sa-elv)
;    (sa-elv)
;    (s-a-elevation)
;    (sa-elv-check) -->
;      (call display rega-sr-elv-again-msg)
;      (modify 3 ^1 sa-elv)
;      (remove 4 5)
;      )
;;
;;  ------------------------------------------------------------------
;;   COORDINATE MAP AND PHOTOS
;;  ------------------------------------------------------------------
;;
;(p rega-map-photo-rule
;    (goal rega)
;    (rega map-photo) -->
;      (call display rega-map-photo-msg)
;      (modify 2 ^2 q75)
;      (call pause)
;      )
;;
;;    50 WRITE(LU(2),55)
(setq rega-map-photo-msg
      '(t t 5 " Now take a few minutes to locate the points" t
        5 "of minimum and maximum terrain elevations of the" t
        5 "'study area' (determined from the map) on the photos." t
        t 5 "In a similar manner locate the minimum and maximum" t
        5 "terrain elevations of the 'study region' on the photos" t
        5 "if your photos extend to these points." t t
        " I'll Wait." t t))
;;       PAUSE 4
;;
;;  ******************************************************************
;;   BRANCH TO TERRAIN LANDFORM GROUP APPROPRIATE FOR MINIMUM
;;   REGIONAL ELEVATION - MINR
;;  ******************************************************************
;;
;;    75 IF(MINR.GE.20)GO TO 200
;;
;;  ------------------------------------------------------------------
;;   QUESTIONS RELATED TO EXISTENCE OF SHORELINE IN REGION OR
;;    STUDY AREA
```

```
;;  ------------------------------------------------------------------
;;
;(p rega-q75-rule
;    (goal rega)
;    (rega q75)
;    (s-r-elevation < 20)
;    (s-r-coastal-shoreline unasked) -->
;      (call display rega-shore-ques-1)
;      (call yes-no s-r-coastal-shoreline)
;      (modify 2 ^2 shore)
;      (remove 4)
;      )
;;
;;        WRITE(LU(2),80)
(setq rega-shore-ques-1
      '(t t 5 "Does an ocean shoreline occur in the" t
        5 "study region?" t t))
;;        READ(LU(1),*)IRC
;;
;;  ------------------------------------------------------------------
;;   IF IRC=0, THEN BY DEFINITION ISC=0 & GO TO TIDAL RIVER QUESTION.
;;  ------------------------------------------------------------------
;;
;(p rega-sr-no-shore-rule
;    (goal rega)
;    (rega shore)
;    (s-r-coastal-shoreline nil)
;    (s-a-coastal-shoreline) -->
;      (modify 2 ^2 sr-tidal-river)
;      (modify 4 ^2 nil rega-sr-no-shore-rule)
;      )
;;
;;        IF(IRC.NE.1)GO TO 90
;;
;;  ------------------------------------------------------------------
;;   IF MINS.GE.20', THEN COASTLINE CANNOT EXIST IN STUDY AREA
;;  ------------------------------------------------------------------
;;
;(p rega-sa-no-shore-rule
;    (goal rega)
;    (rega shore)
;    (s-a-elevation >= 20)
;    (s-a-coastal-shoreline unasked) -->
;      (modify 2 ^2 sr-tidal-river)
;      (modify 4 ^2 nil rega-sa-no-shore-rule)
;      )
;;
;;        IF(MINS.GE.20)GO TO 90
;;        WRITE(LU(2),85)
;;
;(p rega-sa-shore-rule
;    (goal rega)
;    (rega shore)
;    (s-r-coastal-shoreline t)
;    (s-a-coastal-shoreline unasked) -->
;      (call display rega-shore-ques-2)
;      (call yes-no s-a-coastal-shoreline)
;      (remove 4)
;      )
```

```
;;
(setq rega-shore-ques-2
      '(t t 5 "Does the ocean shoreline occur in the" t
        5 "study area?" t t))
;;      READ(LU(1),*)ISC
;;
;; ---------------------------------------------------------------------
;;  WE HAVE DETERMINED THAT THE SHORELINE EXISTS IN THE REGION AND/OR
;;  THE STUDY AREA. REMEMBER THESE TWO CASES: IRC=1 & ISC=1 OR
;;  IRC=1 & ISC=0. NOW YOU CAN ASK QUESTIONS ABOUT THE APPROPRIATE
;;  CASE.
;; ---------------------------------------------------------------------
;;
;;   86 IF(IRC.EQ.1.AND.ISC.EQ.1)GO TO 88
;;      WRITE(LU(2),87)
;;
;(p rega-sr-shore-rule
;   (goal rega)
;   (rega shore)
;   (s-r-coastal-shoreline t)
;   (s-a-coastal-shoreline nil)
;   (s-r-coastal-shoreline-ext unasked) -->
;     (call display rega-sr-shore-ext-ques)
;     (call yes-no s-r-coastal-shoreline-ext)
;     (modify 2 ^2 sr-tidal-river)
;     (remove 5)
;     )
;;
(setq rega-sr-shore-ext-ques
      '(t t 5 "You have indicated the existence of a coastal" t
         5 "shore line in the study region but not in the study" t
         5 "area. Now check the map to determine if some coastal" t
         5 "features might extend into the study area from the" t
         5 "shore line. In general, these features will have" t
         5 "surface elevations <20' unless they are ridges that" t
         5 "parallel the coast." t t
         5 "   Do you believe that the coastal features extend into" t
         5 "the study area?" t t))
;;      READ(LU(1),*)IRCE
;;      GO TO 90
;;
;;   88 WRITE(LU(2),89)
;;
;(p rega-sa-shore-rule-2
;   (goal rega)
;   (rega shore)
;   (s-r-coastal-shoreline t)
;   (s-a-coastal-shoreline t)
;   (s-a-coastal-shoreline-ext unasked) -->
;     (call display rega-sa-shore-ext-ques)
;     (call yes-no s-a-coastal-shoreline-ext)
;     (modify 2 ^2 sr-tidal-river)
;     (remove 5)
;     )
;;
(setq rega-sa-shore-ext-ques
      '(t t 5 "You have indicated the existance of coastline in" t
         5 "the study region and the study area. Now check the" t
         5 "map to determine if the coastal features exist only" t
```

```
          5 "at the shoreline or if they may extend inland.  In" t
          5 "general, these features will have surface elevations" t
          5 "<20' unless they are ridges that parallel the coast." t t
          5 "   Do you believe that the coastal features exist beyond" t
          5 "the shoreline?" t t))
;;       READ(LU(1),*)ISCE
;;
;; ---------------------------------------------------------------
;;      TIDAL RIVER IN REGION OR STUDY AREA?
;; ---------------------------------------------------------------
;;
;(p rega-tidal-river-rule
;    (goal rega)
;    (rega sr-tidal-river)
;    (s-r-tidal-river unasked)-->
;      (call display rega-tidal-river-q-1)
;      (remove 3)
;      (call yes-no s-r-tidal-river)
;      )
;;
;;    90 WRITE(LU(2),95)
(setq rega-tidal-river-q-1
      '(t t 5 "Does a tidal river occur in the study region?" t t))
;;       READ(LU(1),*)IRTR
;;
;; ---------------------------------------------------------------
;;  IF NO TIDAL RIVER IN REGION, THEN BY DEFINITION NONE IN STUDY AREA
;; ---------------------------------------------------------------
;;
;(p rega-sa-no-tidal-river-rule
;    (goal rega)
;    (rega sr-tidal-river)
;    (s-r-tidal-river nil)
;    (s-a-tidal-river unasked) -->
;      (modify 2 ^2 no-coast-no-tr-low)
;      (modify 4 ^2 nil rega-sa-no-tidal-river-rule)
;      )
;;
;;       IF(IRTR.NE.1)GO TO 110
;;
;; ---------------------------------------------------------------
;;  CHECK FOR ISLANDS IN TIDAL RIVER
;; ---------------------------------------------------------------
;;
;(p rega-tidal-river-islands-rule
;    (goal rega)
;    (rega sr-tidal-river)
;    (s-r-tidal-river t)
;    (s-r-tidal-river-islands unasked)-->
;      (call display rega-sr-tr-island-q)
;      (remove 4)
;      (call yes-no s-r-tidal-river-islands)
;      )
;;
;;       WRITE(LU(2),94)
(setq rega-sr-tr-island-q
      '(t t 5 "Does the map indicate one or more islands" t
        5 "in the tidal river outside the study area?" t t))
;;       READ(LU(1),*)IRTRI
```

```
;;
;; ------------------------------------------------------------------
;;  CHECK FOR ROCK IN RIVER
;; ------------------------------------------------------------------
;;
;(p rega-tidal-river-rock-rule
;    (goal rega)
;    (rega sr-tidal-river)
;    (s-r-tidal-river t)
;    (s-r-tidal-river-rock unasked)-->
;      (call display rega-sr-tr-rock-q)
;      (remove 4)
;      (call yes-no s-r-tidal-river-rock)
;      )
;;
;;       WRITE(LU(2),96)
(setq rega-sr-tr-rock-q
      '(t t 5 "Does the map indicate rocks in the tidal" t
        5 "river outside the study area?" t t))
;;       READ(LU(1),*)IRTRR
;;
;; ------------------------------------------------------------------
;;  CHECK FOR MUD FLATS IN RIVER
;; ------------------------------------------------------------------
;;
;(p rega-tidal-river-mud-rule
;    (goal rega)
;    (rega sr-tidal-river)
;    (s-r-tidal-river t)
;    (s-r-tidal-river-mud unasked)-->
;      (call display rega-sr-tr-mud-q)
;      (remove 4)
;      (call yes-no s-r-tidal-river-mud)
;      )
;;
;;       WRITE(LU(2),97)
(setq rega-sr-tr-mud-q
      '(t t 5 "Does the map indicate any mud flats in the" t
        5 "tidal river outside the study area?" t t))
;;       READ(LU(1),*)IRTRM
;;
;; ------------------------------------------------------------------
;;  CHECK FOR SAND BARS IN RIVERS
;; ------------------------------------------------------------------
;;
;(p rega-tidal-river-sand-rule
;    (goal rega)
;    (rega sr-tidal-river)
;    (s-r-tidal-river t)
;    (s-r-tidal-river-sand unasked)-->
;      (call display rega-sr-tr-sand-q)
;      (remove 4)
;      (call yes-no s-r-tidal-river-sand)
;      )
;;
;;
;;       WRITE(LU(2),98)
(setq rega-sr-tr-sand-q
      '(t t 5 "Does the map indicate any sand bars in the tidal" t
```

```
          5 "river outside the study area?" t t))
;;        READ(LU(1),*)IRTRS
;;
;(p rega-sr-tr-to-sa-tr-rule-1
;    (goal rega)
;    (s-a-elevation < 20)
;    (s-r-tidal-river t)
;    (rega sr-tidal-river) -->
;      (modify 4 ^2 sa-tidal-river)
;      )
;;
;(p rega-sr-tr-to-sa-tr-rule-2
;    (goal rega)
;    (s-a-elevation < 20)
;    (s-r-tidal-river nil)
;    (rega sr-tidal-river)
;    (s-a-tidal-river unasked) -->
;      (modify 4 ^2 sa-tidal-river)
;      (modify 5 ^2 nil rega-sr-tr-to-sa-tr-rule-2)
;      )
;;
;(p rega-sr-tr-to-not-sa-tr-rule
;    (goal rega)
;    (s-a-elevation >= 20)
;    (s-r-tidal-river <> unasked)
;    (rega sr-tidal-river) -->
;      (modify 4 ^2 sa-kill)
;      )
;;
;(p rega-sa-tr-kill-rule
;    (goal rega)
;    (rega sa-kill)
;    (s-a-tidal-river)
;    (s-a-tidal-river-islands)
;    (s-a-tidal-river-rock)
;    (s-a-tidal-river-mud)
;    (s-a-tidal-river-sand) -->
;      (modify 2 ^2 low-qs)
;      (modify 3 ^2 nil rega-sa-tr-kill-rule)
;      (modify 4 ^2 nil rega-sa-tr-kill-rule)
;      (modify 5 ^2 nil rega-sa-tr-kill-rule)
;      (modify 6 ^2 nil rega-sa-tr-kill-rule)
;      (modify 7 ^2 nil rega-sa-tr-kill-rule)
;      )
;;
;; ------------------------------------------------------------------
;;  NO TIDAL RIVER IN STUDY AREA IF MINS.GE.20'
;; ------------------------------------------------------------------
;;
;(p rega-sa-tidal-river-rule
;    (goal rega)
;    (rega sa-tidal-river)
;    (s-a-tidal-river unasked) -->
;      (call display rega-sa-tidal-river-q)
;      (call yes-no s-a-tidal-river)
;      (remove 3)
;      )
;;
;;        IF(MINS.GE.20)GO TO 150
```

```
;;       WRITE(LU(2),99)
(setq rega-sa-tidal-river-q
      '(t t 5 "Does a tidal river occur in the study area?" t t))
;;       READ(LU(1),*)ISTR
;;
;;       IF(ISTR.NE.1)GO TO 106
;;
;; ------------------------------------------------------------------
;;  CHECK FOR ISLANDS IN THE STUDY AREA PORTION OF THE TIDAL RIVER
;; ------------------------------------------------------------------
;;
;(p rega-sa-tidal-river-island-rule
;    (goal rega)
;    (rega sa-tidal-river)
;    (s-a-tidal-river t)
;    (s-a-tidal-river-islands unasked) -->
;      (call display rega-sa-tidal-river-islands-q)
;      (call yes-no s-a-tidal-river-islands)
;      (remove 4)
;      )
;;
;;       WRITE(LU(2),100)
(setq rega-sa-tidal-river-islands-q
      '(t t 5 "Does the map indicate one or more islands" t
        5 "in the study area portion of the tidal river?" t t))
;;       READ(LU(1),*)ISTRI
;;
;; ------------------------------------------------------------------
;;  CHECK FOR ROCK IN STUDY AREA PORTION OF RIVER
;; ------------------------------------------------------------------
;;
;(p rega-sa-tidal-river-rock-rule
;    (goal rega)
;    (rega sa-tidal-river)
;    (s-a-tidal-river t)
;    (s-a-tidal-river-rock unasked) -->
;      (call display rega-sa-tidal-river-rock-q)
;      (call yes-no s-a-tidal-river-rock)
;      (remove 4)
;      )
;;
;;       WRITE(LU(2),101)
(setq rega-sa-tidal-river-rock-q
      '(t t 5 "Does the map indicate rocks in the study area" t
        5 "portion of tidal river channel?" t t))
;;       READ(LU(1),*)ISTRR
;;
;; ------------------------------------------------------------------
;;  CHECK FOR MUD FLATS IN STUDY AREA PORTION OF RIVER
;; ------------------------------------------------------------------
;;
;(p rega-sa-tidal-river-mud-rule
;    (goal rega)
;    (rega sa-tidal-river)
;    (s-a-tidal-river t)
;    (s-a-tidal-river-mud unasked) -->
;      (call display rega-sa-tidal-river-mud-q)
;      (call yes-no s-a-tidal-river-mud)
;      (remove 4)
```

```
;       )
;;
;;        WRITE(LU(2),102)
(setq rega-sa-tidal-river-mud-q
      '(t t 5 "Does the map indicate any mud flats in study" t
        5 "area portion of a tidal river?" t t))
;;        READ(LU(1),*)ISTRM
;;
;; ---------------------------------------------------------------
;;  CHECK FOR SAND BARS IN STUDY AREA PORTION OF RIVER
;; ---------------------------------------------------------------
;;
;(p rega-sa-tidal-river-sand-rule
;    (goal rega)
;    (rega sa-tidal-river)
;    (s-a-tidal-river t)
;    (s-a-tidal-river-sand unasked) -->
;      (call display rega-sa-tidal-river-sand-q)
;      (call yes-no s-a-tidal-river-sand)
;      (remove 4)
;      )
;;
;;        WRITE(LU(2),103)
(setq rega-sa-tidal-river-sand-q
      '(t t 5 "Does the map indicate any sand bars in study" t
        5 "area portion of a tidal river?" t t))
;;        READ(LU(1),*)ISTRS
;;
;; ---------------------------------------------------------------
;;  QUESTIONS RELATED TO TIDAL RIVER GO HERE, REMEMBER TWO CASES:
;;    (1) IRTR=1 & ISTR=1 AND (2) IRTR=1 & ISTR=0
;; ---------------------------------------------------------------
;;
;(p rega-sa-tidal-river-ext-rule-1
;    (goal rega)
;    (rega sa-tidal-river)
;    (s-r-tidal-river t)
;    (s-a-tidal-river nil)
;    (s-r-tidal-river-ext unasked) -->
;      (call display rega-sr-tr-no-sa-tr-msg)
;      (call yes-no s-r-tidal-river-ext)
;      (remove 5)
;      )
;;
;;  106 IF(IRTR.EQ.1.AND.ISTR.EQ.1)GO TO 108
;;        WRITE(LU(2),107)
(setq rega-sr-tr-no-sa-tr-msg
      '(t t 5 "You have indicated that a tidal river exists in" t
         5 "the study region but not in the study area. Now check" t
         5 "the map to see if some of the features associated with" t
         5 "the tidal river might extend into the study area. In" t
         5 "general, these features will be salt marshes adjacent" t
         5 "to rivers and have elevations only slightly higher" t
         5 "than the river." t t
         5 "   Do you believe that these features exist?" t t))
;;        READ(LU(1),*)IRTRE
;;        GO TO 150
;;
;(p rega-sa-tidal-river-ext-rule-2
```

```
;    (goal rega)
;    (rega sa-tidal-river)
;    (s-r-tidal-river t)
;    (s-a-tidal-river t)
;    (s-a-tidal-river-ext unasked) -->
;      (call display rega-sr-tr-sa-tr-msg)
;      (call yes-no s-a-tidal-river-ext)
;      (remove 5)
;      )
;;
;;   108 WRITE(LU(2),109)
(setq rega-sr-tr-sa-tr-msg
      '(t t 5 "You have indicated that a tidal river exists in" t
         5 "the study region and in the study area. Now check the" t
         5 "map to see if the features associated with the tidal" t
         5 "river in the study area exist only in the vicinity of" t
         5 "the river or if they extend away from the river. In" t
         5 "general, these features will be salt marsh areas" t
         5 "extending from the river and have elevations only" t
         5 "slightly higher than the river in the area." t t
         5 "   Do you believe that these features exist?" t t))
;;       READ(LU(1),*)ISTRE
;;       GO TO 150
;;
;(p rega-sa-tidal-river-done-rule
;    (goal rega)
;    (rega sa-tidal-river) -->
;       (modify 2 ^2 low-qs)
;      )
;;
;;  ------------------------------------------------------------------
;;   WHEN NEITHER COAST LINE OR TIDAL RIVER OCCURS IN A REGION WITH
;;   ELEVATION <20' WE MUST LOOK FOR REASON.
;;  ------------------------------------------------------------------
;;
;(p no-coast-no-tr-low-rule
;    (goal rega)
;    (rega no-coast-no-tr-low)
;    (s-r-low-coastal-shoreline unasked) -->
;      (remove 3)
;      (call display rega-no-c-no-tr-low-msg-1)
;      (call display rega-no-c-no-tr-low-msg-2)
;      (call yes-no s-r-low-coastal-shoreline)
;      )
;;
;(p no-coast-no-tr-low-done-rule-1
;    (goal rega)
;    (rega no-coast-no-tr-low)
;    (s-r-low-coastal-shoreline t) -->
;      (modify 2 ^2 low-qs)
;      )
;;
;;   110 WRITE(LU(2),111)
(setq rega-no-c-no-tr-low-msg-1
      '(t t 5 "You have indicated that the study region has" t
         5 "neither a coastline or a tidal river, yet the" t
         5 "minimum elevation in the region is <20'. We must" t
         5 "determine what natural feature creates this" t
         5 "condition." t t))
```

```
;;
;; ------------------------------------------------------------------
;;  COULD REGIONAL MINIMUM BE ASSOCIATED WITH A COASTLINE JUST
;;  OUTSIDE THE STUDY REGION?
;; ------------------------------------------------------------------
;;
;;       WRITE(LU(2),113)
(setq rega-no-c-no-tr-low-msg-2
      '(t t 5 "Do you believe that the regional low is" t
        5 "associated with a coastline which exists just beyond" t
        5 "the study region?" t t))
;;       READ(LU(1),*)IRLC
;;       IF(IRLC.EQ.1)GO TO 127
;;
;; ------------------------------------------------------------------
;;  COULD REGIONAL MINIMUM BE ASSOCIATED WITH A TIDAL RIVER JUST
;;  OUTSIDE THE STUDY REGION?
;; ------------------------------------------------------------------
;;
;(p no-coast-no-tr-low-tr-hide-rule
;   (goal rega)
;   (rega no-coast-no-tr-low)
;   (s-r-low-coastal-shoreline nil)
;   (s-r-low-tidal-river-ext unasked) -->
;     (remove 4)
;     (call display rega-no-c-no-tr-low-msg-3)
;     (call yes-no s-r-low-tidal-river-ext)
;     )
;;
;(p no-coast-no-tr-low-done-rule-2
;   (goal rega)
;   (rega no-coast-no-tr-low)
;   (s-r-low-tidal-river-ext t) -->
;     (modify 2 ^2 low-qs)
;     )
;;
;;       WRITE(LU(2),115)
(setq rega-no-c-no-tr-low-msg-3
      '(t t 5 "Do you believe that the regional low is" t
        5 "associated with a tidal river that exists just beyond" t
        5 "the study region?" t t))
;;       READ(LU(1),*)IRLTR
;;       IF(IRLTR.EQ.1)GO TO 127
;;
;; ------------------------------------------------------------------
;;  COULD REGIONAL MINIMUM OCCUR IN ALLUVIAL CHANNEL <20'?
;; ------------------------------------------------------------------
;;
;(p no-coast-no-tr-low-ac-rule
;   (goal rega)
;   (rega no-coast-no-tr-low)
;   (s-r-low-tidal-river-ext nil)
;   (s-r-low-alluvial-channel unasked) -->
;     (remove 4)
;     (call display rega-no-c-no-tr-low-msg-4)
;     (call yes-no s-r-low-alluvial-channel)
;     )
;;
;(p no-coast-no-tr-low-done-rule-3
```

```
;    (goal rega)
;    (rega no-coast-no-tr-low)
;    (s-r-low-alluvial-channel t) -->
;      (modify 2 ^2 low-qs)
;      )
;;
;;   120 WRITE(LU(2),121)
(setq rega-no-c-no-tr-low-msg-4
      '(t t 5 "Does the minimum regional elevation occur" t
         5 "in a dry watercourse or active alluvial stream or" t
         5 "river channel?" t t))
;;       READ(LU(1),*)IRLAC
;;       IF(IRLAC.EQ.1)GO TO 127
;;
;; ---------------------------------------------------------------
;;   IF NONE OF THE ABOVE, ASK FOR HELP.
;; ---------------------------------------------------------------
;;
;(p no-coast-no-tr-low-done-rule-4
;    (goal rega)
;    (rega no-coast-no-tr-low)
;    (s-r-low-alluvial-channel nil) -->
;      (call display rega-no-c-no-tr-low-msg-5)
;      (call display rega-no-c-no-tr-low-msg-6)
;      (modify 2 ^2 low-qs)
;      )
;;
;;       WRITE(LU(2),123)
(setq rega-no-c-no-tr-low-msg-5
      '(t t 5 "You have indicated that neither coastline nor" t
        5 "tidal river landform groups exist in the study region" t
        5 "and you do not believe that these features exist" t
        5 "outside the region in a manner such that their" t
        5 "associated features could extend into the study" t
        5 "region and explain the regional minimum elevation." t
        5 "Further you have indicated that the regional minimum" t
        5 "does not appear to be related to an alluvial channel." t t
        5 "We have exhausted the available alternatives, thus I" t
        5 "suggest you consult your supervisor." t t))
;       WRITE(LU(2),124)
(setq rega-no-c-no-tr-low-msg-6
      '(t t 5 "You will be allowed to continue" t
        5 "Please use caution." t t))
;;       READ(LU(1),*)IHELP1
;;       IF(IHELP1.EQ.0)GO TO 127
;;       WRITE(LU(2),125)
(setq rega-fools-help-msg
      '(" DEAD IN THE WATER (and you expected help)" t t t))
;;       GO TO 1000
;;
;; ---------------------------------------------------------------
;;   QUESTIONS RELATED TO STUDY AREA MINIMUM <20' WITH OCCURRENCE NOT
;;    ON COASTLINE OR TIDAL RIVER
;; ---------------------------------------------------------------
;;
;(p rega-low-qs-start-rule
;    (goal rega)
;    (rega low-qs)
;    (s-r-coastal-shoreline nil)
```

```
;    (s-r-tidal-river nil)
;    (s-a-elevation < 20) -->
;      (call display rega-120-msg-1)
;      (make low-qs)
;      )
;;
;(p rega-low-qs-skip-to-mid-rule
;    (goal rega)
;    (rega low-qs) -->
;      (make low-150-qs)
;      )
;;
;;   127 IF(MINS.GE.20)GO TO 160
;;       WRITE(LU(2),129)
(setq rega-120-msg-1
      '(t t 5 "It has been indicated that the study region" t
        5 "contains neither a coastline nor a tidal river, yet" t
        5 "the minimum elevation in the study area is <20'. We" t
        5 "must determine which natural feature causes this" t
        5 "condition." t t))
;;
;;       IF(IRLC.EQ.0)GO TO 133
;;
;; ---------------------------------------------------------------
;;  COULD THE STUDY AREA MINIMUM BE ASSOCIATED WITH A COASTLINE
;;   JUST OUTSIDE THE STUDY REGION SUCH THAT FEATURES OF THE
;;   COASTLINE EXTEND INTO THE STUDY AREA?
;; ---------------------------------------------------------------
;;
;(p rega-low-qs-sa-min-coast-rule
;    (goal rega)
;    (rega low-qs)
;    (low-qs)
;    (s-r-low-coastal-shoreline t)
;    (s-a-low-coastal-shoreline unasked) -->
;      (call display rega-120-msg-2)
;      (call yes-no s-a-low-coastal-shoreline)
;      (remove 5)
;      )
;;
;;       WRITE(LU(2),131)
(setq rega-120-msg-2
      '(t t 5 "You have said you believe the regional low is" t
        5 "associated with a coastline beyond the study region" t
        5 "boundary. Do you think that the study area low is" t
        5 "also associated with this coastline?" t t))
;;       READ(LU(1),*)ISLC
;;       GO TO 135
;;
;; ---------------------------------------------------------------
;;  COULD THE STUDY AREA MINIMUM BE ASSOCIATED WITH A TIDAL RIVER
;;   JUST OUTSIDE THE STUDY REGION SUCH THAT FEATURES OF THE
;;   TIDAL RIVER EXTEND INTO THE STUDY AREA?
;; ---------------------------------------------------------------
;;
;(p rega-low-qs-sa-min-tidal-river-rule
;    (goal rega)
;    (rega low-qs)
;    (low-qs)
```

```
;    (s-r-low-coastal-shoreline nil)
;    (s-a-low-tidal-river unasked) -->
;      (call display rega-120-msg-3)
;      (call yes-no s-a-low-tidal-river)
;      (remove 5)
;      )
;;
;;   133 IF(IRLTR.EQ.0)GO TO 150
;;       WRITE(LU(2),134)
(setq rega-120-msg-3
      '(t t 5 "You believe the regional low is associated with" t
        5 "a tidal river beyond the study region boundary." t
        5 "Do you believe that the study area low is also" t
        5 "associated with the tidal river?" t t))
;;       READ(LU(1),*)ISLTR
;;   135 IF(IRLC.NE.0.OR.IRLTR.NE.0)GO TO 150
;;
;; ------------------------------------------------------------
;;  COULD STUDY AREA MINIMUM OCCUR IN AN ALLUVIAL CHANNEL?
;; ------------------------------------------------------------
;;
;(p rega-low-qs-sa-min-alluviel-channel-rule
;    (goal rega)
;    (rega low-qs)
;    (low-qs)
;    (s-r-low-coastal-shoreline <> t)
;    (s-r-low-tidal-river <> t)
;    (s-a-low-alluviel-channel unasked) -->
;      (call display rega-120-msg-4)
;      (call yes-no s-a-low-alluviel-channel)
;      (remove 6)
;      )
;;
;;       WRITE(LU(2),137)
(setq rega-120-msg-4
      '(t t 5 "Does the minimum study area elevation occur" t
        5 "in a dry watercourse or active alluvial stream or" t
        5 "river channel?" t t))
;;       READ(LU(1),*)ISLAC
;;
;; ------------------------------------------------------------
;;  IF MAXS.LT.20, THEN ALL OF STUDY AREA BELOW 20'
;; ------------------------------------------------------------
;;
;(p rega-low-qs-done-rule
;    (goal rega)
;    (rega low-qs)
;    (low-qs) -->
;      (modify 3 ^1 low-150-qs)
;      )
;;
;;   150 IF(MAXS.GE.20)GO TO 160
;;
;; ------------------------------------------------------------
;;  QUESTIONS RELATED TO STUDY AREA MAXIMUM <20'
;; ------------------------------------------------------------
;;
;(p rega-low-qs-sa-max-coast-rule
;    (goal rega)
```

```
;    (rega low-qs)
;    (low-150-qs)
;    (s-a-elevation {}: < 20)
;    (s-a-max-coastal-shoreline unasked) -->
;      (call display rega-1120-msg-1)
;      (call yes-no s-a-max-coastal-shoreline)
;      (remove 5)
;      )
;;
;;       WRITE(LU(2),151)
(setq rega-1120-msg-1
      '(t t 5 "We now want to associate the maximum elevation" t
        5 "in the study area with either coastal features or" t
        5 "features associated with tidal rivers." t t
;
        5 "Is the maximum elevation in the study area" t
        5 "associated with coastal features?" t t))
;;       READ(LU(1),*)ISMC
;;       IF(ISMC.EQ.1)GO TO 160
;;
;(p rega-low-qs-sa-max-tidal-river-rule
;    (goal rega)
;    (rega low-qs)
;    (low-150-qs)
;    (s-a-elevation {}: < 20)
;    (s-a-max-coastal-shoreline nil)
;    (s-a-max-tidal-river unasked) -->
;      (call display rega-1120-msg-2)
;      (call yes-no s-a-max-tidal-river)
;      (remove 6)
;      )
;;
;;       WRITE(LU(2),153)
(setq rega-1120-msg-2
      '(t t 5 "Is the maximum elevation in the study" t
        5 "area associated with tidal rivers such as a marsh" t
        5 "or levee?" t t))
;;       READ(LU(1),*)ISMTR
;;       IF(ISMTR.EQ.1)GO TO 160
;;
;(p rega-low-qs-sa-max-un-associated-rule
;    (goal rega)
;    (rega low-qs)
;    (low-150-qs)
;    (s-a-elevation {}: < 20)
;    (s-a-max-tidal-river nil)
;    (s-a-max-un-associated unasked) -->
;      (call display rega-1120-msg-3)
;      (call yes-no s-a-max-un-associated)
;      (remove 6)
;      )
;;
;;       WRITE(LU(2),155)
(setq rega-1120-msg-3
      '(t t 5 "Is it difficult to associate the study area" t
        5 "maximum with any particular terrain feature?" t t))
;;       READ(LU(1),*)ISMZ
;;
;(p rega-low-150-qs-done-rule
```

```
;    (goal rega)
;    (rega low-qs)
;    (low-150-qs) -->
;      (modify 3 ^1 mid-qs)
;      )
;;
;; ------------------------------------------------------------------
;;  IF MAXR.LT.20, THEN ALL REGION BELOW 20'
;; ------------------------------------------------------------------
;;
;(p rega-low-qs-mid-done-rule
;    (goal rega)
;    (rega low-qs)
;    (mid-qs) -->
;      (modify 2 ^2 sr-low)
;      (remove 3)
;      )
;;
;;  160 IF(MAXR.GE.20)GO TO 200
;;
;; ------------------------------------------------------------------
;;  QUESTIONS RELATED TO STUDY REGION MAXIMUM <20'
;; ------------------------------------------------------------------
;;
;;       CONTINUE
;;
;; ------------------------------------------------------------------
;;  ALL LANDFORMS <20', GO TO REGIONAL DRAINAGE SECTION
;; ------------------------------------------------------------------
;;
;;       GO TO 500
;;
;; ********************************************************************
;;
;;  LOW TERRACE IN STUDY REGION OR STUDY AREA?
;;
;; ********************************************************************
;;
;;  200 IF(MINR.LT.100.AND.MAXR.GT.20)GO TO 205
;;       GO TO 300
;;
;(p rega-sr-terrace-start-rule
;    (goal rega)
;    (rega sr-low) -->
;      (modify 2 ^2 terraces)
;      (make terraces sr-low)
;      )
;;
;; ********************************************************************
;;  QUESTIONS RELATED TO LOW TERRACES IN STUDY REGION
;; ********************************************************************
;;
;(p rega-sr-low-terrace-dont-ask-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sr-low)
;    (s-r-low-terraces unasked) -->
;      (modify 3 ^2 sr-mid)
;      )
```

```
;;
;(p rega-sr-low-terrace-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sr-low)
;    (s-r-elevation < 100 > 20)
;    (s-r-low-terraces unasked) -->
;      (call display rega-sr-ltr-msg-1)
;      (call display rega-sr-ltr-msg-2)
;      (call pause)
;      (call display rega-sr-ltr-msg-3)
;      (call display rega-sr-ltr-msg-4)
;      (call yes-no s-r-low-terraces)
;      (remove 5)
;      )
;;
;;   205 WRITE(LU(2),206)
(setq rega-sr-ltr-msg-1
      '(t t 5 "Now check your map sheet of the study region" t
        5 "and study area for elevated land between elevations of" t
        5 "approximately 20' and 100' which have wide contour" t
        5 "spacings indicative of generally level surfaces." t t))
;;
;;   208 WRITE(LU(2),210)
(setq rega-sr-ltr-msg-2
      '(t t 5 "Please check your map sheet in detail!" t t))
;;      PAUSE 5
;;
;;      WRITE(LU(2),215)
(setq rega-sr-ltr-msg-3
      '(t t " You will now be asked a few questions concerning" t
        " the land areas between elevations of" t
        " approximately 20' and 100' in the study region and study" t
        " area." t t))
;;
;;      WRITE(LU(2),220)
(setq rega-sr-ltr-msg-4
      '(t t 5 "Are there elevated and generally level land surfaces" t
        5 "with elevations between 20' and 100' in the" t
        5 "study region?" t t))
;;      READ(LU(1),*)IRT10
;;
;(p rega-sr-no-low-terrace-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sr-low)
;    (s-r-low-terraces nil) -->
;      (modify 3 ^2 sa-low)
;      )
;;
;;      IF(IRT10.NE.1)GO TO 250
;;
;(p rega-sr-low-terrace-coast-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sr-low)
;    (s-r-low-terraces t)
;    (s-r-coastal-shoreline t)
;    (s-r-low-terraces-near-coast unasked) -->
```

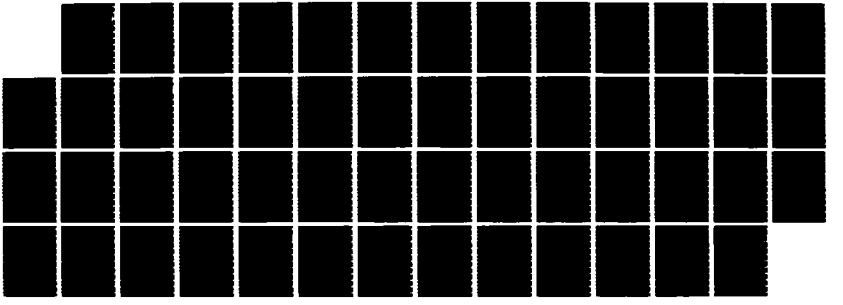
```
;       (call display rega-sr-ltr-msg-5)
;       (call yes-no s-r-low-terraces-near-coast)
;       (remove 6)
;       )
;;
;;        IF(IRC.NE.1)GO TO 230
;;        WRITE(LU(2),225)
(setq rega-sr-ltr-msg-5
      '(t t 5 "Are these surfaces in the vicinity of the" t
        5 "coastline?" t t))
;;        READ(LU(1),*)IRT1C
;;
;(p rega-sr-low-terrace-tidal-river-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sr-low)
;    (s-r-low-terraces t)
;    (s-r-tidal-river t)
;    (s-r-low-terraces-near-tidal-river unasked) -->
;      (call display rega-sr-ltr-msg-6)
;      (call yes-no s-r-low-terraces-near-tidal-river)
;      (remove 6)
;      )
;;
;;   230 IF(IRTR.NE.1)GO TO 250
;;        WRITE(LU(2),231)
(setq rega-sr-ltr-msg-6
      '(t t 5 "Are these surfaces in the vicinity of the" t
        5 "tidal river?" t t))
;;        READ(LU(1),*)IRT1TR
;;
;(p rega-sr-low-terrace-to-sa-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sr-low) -->
;      (modify 3 ^2 sa-low)
;      )
;;
;;  ************************************************************************
;;   QUESTIONS RELATED TO LOW TERRACES IN THE STUDY AREA
;;  ************************************************************************
;;
;(p rega-sa-low-terrace-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sa-low)
;    (s-a-elevation < 100 > 20)
;    (s-r-low-terraces t)
;    (s-a-low-terraces unasked) -->
;      (call display rega-sa-ltr-msg-1)
;      (call yes-no s-a-low-terraces)
;      (remove 6)
;      )
;;
;(p rega-sa-low-terrace-no-sr-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sa-low)
;    (s-r-low-terraces nil)
```

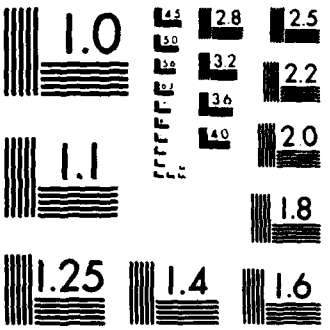
```
;    (s-a-low-terraces unasked) -->
;      (modify 5 ^2 nil rega-sa-low-terrace-no-sr-rule)
;      )
;;
;;   250 IF(MINS.LT.100.AND.MAXS.GT.20)GO TO 252
;;       GO TO 300
;;
;;   252 WRITE(LU(2),253)
(setq rega-sa-ltr-msg-1
      '(t t 5 "Do these surfaces exist in the study area?" t t))
;;       READ(LU(1),*)IST1O
;;
;(p rega-sa-low-terrace-coast-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sa-low)
;    (s-a-coastal-shoreline t)
;    (s-a-low-terraces t)
;    (s-a-low-terraces-near-coast unasked) -->
;      (call display rega-sa-ltr-msg-2)
;      (call yes-no s-a-low-terraces-near-coast)
;      (remove 6)
;      )
;;
;;       IF(ISC.NE.1)GO TO 260
;;       WRITE(LU(2),255)
(setq rega-sa-ltr-msg-2
      '(t t 5 "Are these surfaces in the study area located" t
        5 "in the vicinity of the coastline?" t t))
;;       READ(LU(1),*)IST1C
;;
;(p rega-sa-low-terrace-tidal-river-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sa-low)
;    (s-a-tidal-river t)
;    (s-a-low-terraces t)
;    (s-a-low-terraces-near-tidal-river unasked) -->
;      (call display rega-sa-ltr-msg-3)
;      (call yes-no s-a-low-terraces-near-tidal-river)
;      (remove 6)
;      )
;;
;;   260 IF(ISTR.NE.1)GO TO 300
;;       WRITE(LU(2),265)
(setq rega-sa-ltr-msg-3
      '(t t 5 "Are these surfaces in the study area located" t
        5 "in the vicinity of a tidal river?" t t))
;;       READ(LU(1),*)IST1TR
;;
;(p rega-sa-low-terrace-done-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sa-low) -->
;      (modify 3 ^2 sr-mid)
;      )
;;
;;   LAST STATEMENT
;;       IF(MAXR.LT.100)GO TO 500
```

```
;;
;;  ********************************************************************
;;
;;   MID LEVEL TERRACES IN REGION AND STUDY AREA
;;
;;  ********************************************************************
;;
;;   300 IF(MINR.LT.170.AND.MAXR.GT.100)GO TO 305
;;       GO TO 400
;;
;;  --------------------------------------------------------------------
;;   QUESTIONS RELATED TO MID LEVEL TERRACES IN THE STUDY REGION
;;  --------------------------------------------------------------------
;;
;(p rega-sr-mid-terrace-dont-ask-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sr-mid)
;    (s-r-mid-terraces unasked) -->
;      (modify 3 ^2 sr-high)
;      )
;;
;(p rega-sr-mid-terrace-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sr-mid)
;    (s-r-elevation < 170 > 100)
;    (s-r-mid-terraces unasked) -->
;      (call display rega-sr-mtr-msg-1)
;      (call display rega-sr-mtr-msg-2)
;      (call pause)
;      (call display rega-sr-mtr-msg-3)
;      (call display rega-sr-mtr-msg-4)
;      (call yes-no s-r-mid-terraces)
;      (remove 5)
;      )
;;
;;   305 WRITE(LU(2),306)
(setq rega-sr-mtr-msg-1
      '(t t 5 "Now check your map sheet of the study region and" t
        5 "study area for elevated land surfaces between" t
        5 "elevations of approximately 100' and 170' which have" t
        5 "wide contour spacings indicative of generally level" t
        5 "surfaces. Transition slopes from these surfaces to" t
        5 "lower levels will be steep and uniform." t t))
;;   308 WRITE(LU(2),310)
(setq rega-sr-mtr-msg-2
      '(t t 5 "Please check the map sheet in detail!" t t))
;;      PAUSE 6
;;
;;      WRITE(LU(2),315)
(setq rega-sr-mtr-msg-3
      '(t t 5 "You will now be asked a few questions about the" t
        5 "elevated land areas between elevations of approximately" t
        5 "100' and 170' in the study region and study area." t t))
;;      WRITE(LU(2),320)
(setq rega-sr-mtr-msg-4
      '(t t 5 "Are there elevated and generally level land surfaces" t
        5 "with elevations between 100' and 170' in the" t
```

AD-A172 315 CONVERSION OF THE CALAP (COMPUTER AIDED LANDFORM ANALYSIS PROGRAM) PROGRAM FROM FORTRAN TO DUCK(U) JAYCOR ALEXANDRIA VA SEP 86 ETL-0419 DAAK70-82-C-0027 2/2

UNCLASSIFIED F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

```
        5 "study region?" t t))
;;      READ(LU(1),*)IRT20
;;
;(p rega-sr-mid-terrace-to-sa-rule
;   (goal rega)
;   (rega terraces)
;   (terraces sr-mid) -->
;     (modify 3 ^2 sa-mid)
;     )
;;
;;      IF(IRT20.NE.1)GO TO 350
;;  ADD ADDITIONAL STATEMENTS ON MID LEVEL TERRACES IN STUDY REGION
;;      CONTINUE
;;
;; ------------------------------------------------------------------
;;  QUESTIONS RELATED TO MID LEVEL TERRACES IN THE STUDY AREA
;; ------------------------------------------------------------------
;;
;(p rega-sa-mid-terrace-rule
;   (goal rega)
;   (rega terraces)
;   (terraces sa-mid)
;   (s-a-elevation < 170 > 100)
;   (s-r-mid-terraces t)
;   (s-a-mid-terraces unasked) -->
;     (call display rega-sa-mtr-msg-1)
;     (call yes-no s-a-mid-terraces)
;     (remove 6)
;     )
;;
;(p rega-sa-mid-terrace-no-sr-rule
;   (goal rega)
;   (rega terraces)
;   (terraces sa-mid)
;   (s-r-mid-terraces nil)
;   (s-a-mid-terraces unasked) -->
;     (modify 5 ^2 nil rega-sa-mid-terrace-no-sr-rule)
;     )
;;
;;  350 IF(MINS.LT.170.AND.MAXS.GT.100)GO TO 355
;;      GO TO 400
;;  355 WRITE(LU(2),356)
(setq rega-sa-mtr-msg-1
      '(t t 5 "Are there elevated and generally level land surfaces" t
        5 "with elevations between 100' and 170' in the" t
        5 "study area?" t t))
;;      READ(LU(1),*)IST20
;;      IF(IST20.NE.1)GO TO 400
;;  ADD ADDITIONAL STATEMENTS ON MID LEVEL TERRACES IN STUDY AREA
;;      CONTINUE
;;
;(p rega-sa-mid-terrace-done-rule
;   (goal rega)
;   (rega terraces)
;   (terraces sa-mid) -->
;     (modify 3 ^2 sr-high)
;     )
;;
;; ******************************************************************
```

```
;;  HIGH LEVEL TERRACES IN THE STUDY REGION AND STUDY AREA
;; ************************************************************************
;;
;;  400 IF(MINR.LT.270.AND.MAXR.GT.170)GO TO 405
;;      GO TO 500
;;
;(p rega-sr-high-terrace-dont-ask-rule
;   (goal rega)
;   (rega terraces)
;   (terraces sr-high)
;   (s-r-high-terraces unasked) -->
;     (modify 2 ^2 drainage)
;     (remove 3)
;     )
;;
;(p rega-sr-high-terrace-rule
;   (goal rega)
;   (rega terraces)
;   (terraces sr-high)
;   (s-r-elevation < 270 > 170)
;   (s-r-high-terraces unasked) -->
;     (call display rega-sr-htr-msg-1)
;     (call display rega-sr-htr-msg-2)
;     (call pause)
;     (call display rega-sr-htr-msg-3)
;     (call display rega-sr-htr-msg-4)
;     (call yes-no s-r-high-terraces)
;     (remove 5)
;     )
;;
;;  405 WRITE(LU(2),406)
(setq rega-sr-htr-msg-1
      '(t t 5 "Now check your map sheet of the study region" t
        5 "and study area for elevated land areas between" t
        5 "elevations of approximately 170' and 270'. We are" t
        5 "interested in level areas or erroded remnants of once" t
        5 "level surfaces. The slopes from these surfaces will" t
        5 "be steep and uniform." t t))
;;  408 WRITE(LU(2),409)
(setq rega-sr-htr-msg-2
      '(t t 5 "Please check the map sheet in detail!" t t))
;;      PAUSE 7
;;
;; ----------------------------------------------------------------------
;;  QUESTIONS RELATED TO HIGH LEVEL TERRACES IN THE STUDY REGION
;; ----------------------------------------------------------------------
;;
;;      WRITE(LU(2),415)
(setq rega-sr-htr-msg-3
      '(t t 5 "You will now be asked a few questions about the" t
        5 "elevated land areas between elevations of approximately" t
        5 "170' and 270' in the study region and study area." t t))
;;
;;      WRITE(LU(2),420)
(setq rega-sr-htr-msg-4
      '(t t 5 "Did you find elevated land surfaces or" t
        5 "erroded remnants of once level surfaces in the study" t
        5 "region between the elevations of approximately 170'" t
        5 "and 270'?" t t))
```

```
;;       READ(LU(1),*)IRT30
;;       IF(IRT30.NE.1)GO TO 450
;;   ADD ADDITIONAL STATEMENTS FOR HIGH LEVEL TERRACES IN STUDY REGION
;;       CONTINUE
;;
;(p rega-sr-high-terrace-to-sa-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sr-high) -->
;      (modify 3 ^2 sa-high)
;      )
;;
;; ------------------------------------------------------------------
;;  QUESTIONS RELATED TO HIGH LEVEL TERRACES IN THE STUDY AREA
;; ------------------------------------------------------------------
;;
;(p rega-sa-high-terrace-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sa-high)
;    (s-a-elevation < 270 > 170)
;    (s-r-high-terraces t)
;    (s-a-high-terraces unasked) -->
;      (call display rega-sa-htr-msg-1)
;      (call yes-no s-a-high-terraces)
;      (remove 6)
;      )
;;
;(p rega-sa-high-terrace-no-sr-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sa-high)
;    (s-r-high-terraces nil)
;    (s-a-high-terraces unasked) -->
;      (modify 5 ^2 nil rega-sa-high-terrace-no-sr-rule)
;      )
;;
;;   450 IF(MINS.LT.270.AND.MAXS.GT.170)GO TO 455
;;       GO TO 500
;;   455 WRITE(LU(2),460)
(setq rega-sa-htr-msg-1
      '(t t 5 "Are there elevated level areas or erroded remnants of" t
        5 "once level surfaces in the study area between" t
        5 "the elevations of approximately 170' and 270'?" t t))
;;       READ(LU(1),*)IST30
;;       IF(IST30.NE.1)GO TO 500
;;   ADD ADDITIONAL STATEMENTS OF HIGH LEVEL TERRACES IN STUDY AREA
;;       CONTINUE
;;
;(p rega-sa-high-terrace-done-rule
;    (goal rega)
;    (rega terraces)
;    (terraces sa-high) -->
;      (modify 2 ^2 drainage)
;      (remove 3)
;      )
;;
;; ******************************************************************
;;  DRAINAGE IN STUDY REGION AND STUDY AREA
```

```
;;  ************************************************************************
;;
;(p rega-drainage-start-rule
;    (goal rega)
;    (rega drainage)
;    (s-r-rivers unasked) -->
;      (call display rega-sr-drg-msg-1)
;      (call pause)
;      (call display rega-sr-drg-msg-2)
;      (call yes-no s-r-rivers)
;      (remove 3)
;      (make drainage sr-rivers)
;      )
;;
;;   500 WRITE(LU(2),505)
(setq rega-sr-drg-msg-1
      '(t t 5 "Now we are interested in the fresh water rivers" t
        5 "and streams in the study region and study area." t
        5 "Here we arbitrarily define a 'river' as a watercourse" t
        5 "which is indicated by a double line symbol on a" t
        5 "1:50,000 scale topographic map and a 'stream' will be" t
        5 "indicated by a single line symbol. This will avoid the" t
        5 "ambiguities associated with local names. Now scan the" t
        5 "map sheet to determine the general locations of the" t
        5 "rivers and streams." t t
        5 "Please check the map!" t t))
;;       PAUSE 10
;;
;;  ************************************************************************
;;   DRAINAGE CHANNELS IN THE STUDY REGION
;;  ************************************************************************
;;
;;       WRITE(LU(2),510)
(setq rega-sr-drg-msg-2
      '(t t 5 "You will now be asked questions about the" t
        5 "rivers and streams in the study region." t t
        5 "Are there one or more rivers in the study region?" t t))
;;
;(p rega-sr-drain-rule-2
;    (goal rega)
;    (rega drainage)
;    (drainage sr-rivers)
;    (s-r-rivers nil) -->
;      (modify 3 ^2 sr-streams)
;      )
;;
;;       READ(LU(1),*)IRAC
;;       IF(IRAC.NE.1)GO TO 550
;;
;(p rega-sr-drain-riv-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-rivers)
;    (s-r-rivers t)
;    (s-r-meandering-rivers unasked) -->
;      (call display rega-sr-drg-msg-3)
;      (call yes-no s-r-meandering-rivers)
;      (remove 5)
;      )
```

```
;;
;;       WRITE(LU(2),515)
(setq rega-sr-drg-msg-3
      '(t t 5 "Does any river have meanders?" t t))
;;
;(p rega-sr-riv-l-meander-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-rivers)
;    (s-r-meandering-rivers t)
;    (s-r-elevation < 101 )
;    (s-r-low-meandering-rivers unasked) -->
;      (call display rega-sr-drg-msg-4)
;      (call yes-no s-r-low-meandering-rivers)
;      (remove 6)
;      )
;;
;(p rega-sr-riv-h-meander-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-rivers)
;    (s-r-meandering-rivers t)
;    (s-r-elevation > 100 )
;    (s-r-high-meandering-rivers unasked) -->
;      (call display rega-sr-drg-msg-5)
;      (call yes-no s-r-high-meandering-rivers)
;      (remove 6)
;      )
;;
;;       READ(LU(1),*)IRACM
;;       IF(IRACM.NE.1)GO TO 530
;;
;;       IF(MINR.GT.100)GO TO 524
;;       WRITE(LU(2),520)
(setq rega-sr-drg-msg-4
      '(t t 5 "Does any river have meanders below the" t
        5 "elevation of 100'?" t t))
;;       READ(LU(1),*)IRACML
;;
;;       IF(MAXR.LT.100)GO TO 530
;;  524 WRITE(LU(2),525)
(setq rega-sr-drg-msg-5
      '(t t 5 "Does any river have meanders above the " t
        5 "elevation of 100'?" t t))
;;       READ(LU(1),*)IRACMH
;;
;(p rega-sr-riv-meander-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-rivers)
;    (s-r-rapid/fall-rivers unasked)
;    (s-r-deflected-rivers unasked) -->
;      (call display rega-sr-drg-msg-6)
;      (call yes-no s-r-rapid/fall-rivers)
;      (call display rega-sr-drg-msg-7)
;      (call yes-no s-r-deflected-rivers)
;      (remove 4 5)
;      )
;;
```

```
;;   530 WRITE(LU(2),535)
(setq rega-sr-drg-msg-6
      '(t t 5 "Does the map indicate the existance of" t,
        5 "falls or rapids in any study region rivers?" t t))
;;       READ(LU(1),*)IRARF
;;
;;       WRITE(LU(2),540)
(setq rega-sr-drg-msg-7
      '(t t 5 "Does any study region river have" t
        5 "a sharp alignment change as though the water is being" t
        5 "deflected by a resistant material?" t t))
;;       READ(LU(1),*)IRACD
;;       IF(IRACD.NE.1)GO TO 550
;;
;;       IF(MAXR.LT.100)GO TO 550
;;       WRITE(LU(2),545)
;;
;(p rega-sr-riv-sac-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-rivers)
;    (s-r-elevation > 100)
;    (s-r-deflected-rivers t)
;    (s-r-high-deflected-rivers unasked) -->
;      (call display rega-sr-drg-msg-8)
;      (call yes-no s-r-high-deflected-rivers)
;      (remove 6)
;      )
;;
(setq rega-sr-drg-msg-8
      '(t t 5 "Does the sharp alignment change" t
        5 "occur above the elevation of 100'?" t t))
;;
;(p rega-sr-riv-to-str-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-rivers) -->
;      (modify 3 ^2 sr-streams)
;      )
;;
;;       READ(LU(1),*)IRACDH
;;
;(p rega-sr-str-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-streams)
;    (s-r-streams unasked) -->
;      (call display rega-sr-drg-msg-9)
;      (call yes-no s-r-streams)
;      (remove 4)
;      )
;;
;;   550 WRITE(LU(2),555)
(setq rega-sr-drg-msg-9
      '(t t 5 "Are there any streams in the" t
        5 "study region?" t t))
;;
;(p rega-sr-no-str-rule
;    (goal rega)
```

```
;    (rega drainage)
;    (drainage sr-streams)
;    (s-r-streams nil) -->
;      (modify 3 ^2 sa-rivers)
;      )
;;
;;       READ(LU(1),*)IRS
;;       IF(IRS.NE.1)GO TO 600
;;
;(p rega-sr-str-meander-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-streams)
;    (s-r-streams t)
;    (s-r-meandering-streams unasked) -->
;      (call display rega-sr-drg-msg-10)
;      (call yes-no s-r-meandering-streams)
;      (remove 5)
;      )
;;
;;       WRITE(LU(2),560)
(setq rega-sr-drg-msg-10
      '(t t 5 "Are there any meanders along any of" t
        5 "the streams in the study region?" t t))
;;       READ(LU(1),*)IRSM
;;       IF(IRSM.NE.1)GO TO 570
;;
;(p rega-sr-str-h-meander-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-streams)
;    (s-r-meandering-streams t)
;    (s-r-elevation {} > 99)
;    (s-r-high-meandering-streams unasked) -->
;      (call display rega-sr-drg-msg-11)
;      (call yes-no s-r-high-meandering-streams)
;      (remove 6)
;      )
;;
;;       IF(MAXR.LT.100)GO TO 570
;;       WRITE(LU(2),565)
(setq rega-sr-drg-msg-11
      '(t t 5 "Do any meander segments occur above" t
        5 "the elevation of 100'?" t t))
;;       READ(LU(1),*)IRSMH
;;
;(p rega-sr-str-dseg-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-streams)
;    (s-r-streams t)
;    (s-r-distributary-segments unasked) -->
;      (call display rega-sr-drg-msg-12)
;      (call yes-no s-r-distributary-segments)
;      (remove 5)
;      )
;;
;;  570 WRITE(LU(2),575)
(setq rega-sr-drg-msg-12
```

```
       '(t t 5 "Do any of the streams have" t
         5 "distributary segments?" t t))
;;       READ(LU(1),*)IRSD
;;       IF(IRSD.NE.1)GO TO 585
;;
;(p rega-sr-str-h-dseg-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-streams)
;    (s-r-distributary-segments t)
;    (s-r-elevation {} > 99)
;    (s-r-high-distributary-segments unasked) -->
;      (call display rega-sr-drg-msg-13)
;      (call yes-no s-r-high-distributary-segments)
;      (remove 6)
;      )
;;
;;       IF(MAXR.LT.100)GO TO 585
;;       WRITE(LU(2),580)
(setq rega-sr-drg-msg-13
       '(t t 5 "Do any of the distributary segments" t
         5 "occur above the elevation of 100'?" t t))
;;       READ(LU(1),*)IRSDH
;;
;(p rega-sr-str-gradients-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-streams)
;    (s-r-streams t)
;    (s-r-strange-gradients unasked) -->
;      (call display rega-sr-drg-msg-14)
;      (call yes-no s-r-strange-gradients)
;      (remove 5)
;      )
;;
;;  585 WRITE(LU(2),590)
(setq rega-sr-drg-msg-14
       '(t t 5 "Now scan from the low elevations to high" t
         5 "elevations along the streams in the" t
         5 "study region to detect unanticipated gradient" t
         5 "changes.  Do unexpected gradient changes occur?" t t))
;;       READ(LU(1),*)IRSG
;;       IF(IRSG.NE.1)GO TO 600
;;
;(p rega-sr-str-gradients-incised-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sr-streams)
;    (s-r-strange-gradients t)
;    (s-r-strange-gradients-incised unasked) -->
;      (call display rega-sr-drg-msg-15)
;      (call yes-no s-r-strange-gradients-incised)
;      (remove 5)
;      )
;;
;;       WRITE(LU(2),595)
(setq rega-sr-drg-msg-15
       '(t t 5 "Is the stream incised and" t
         5 "the valley walls constricted in the vicinity of the" t
```

```
        5 "gradient change?" t t))
;;      READ(LU(1),*)IRSGI
;;
;(p rega-sr-str-done-rule
;   (goal rega)
;   (rega drainage)
;   (drainage sr-streams) -->
;     (modify 3 ^2 sa-rivers)
;     )
;;
;; ************************************************************************
;;  STUDY AREA DRAINAGE
;; ************************************************************************
;;
;(p rega-sa-drainage-start-rule
;   (goal rega)
;   (rega drainage)
;   (drainage sa-rivers)
;   (s-r-rivers t)
;   (s-a-rivers unasked) -->
;     (call display rega-sa-drg-msg-1)
;     (call yes-no s-a-rivers)
;     (remove 5)
;     )
;;
;(p rega-sa-no-sr-river-rule
;   (goal rega)
;   (rega drainage)
;   (drainage sa-rivers)
;   (s-r-rivers nil)
;   (s-a-rivers unasked) -->
;     (modify 3 ^2 sa-streams)
;     (modify 5 ^2 nil rega-sa-no-sr-river-rule)
;     )
;;
;(p rega-sa-no-sa-river-rule
;   (goal rega)
;   (rega drainage)
;   (drainage sa-rivers)
;   (s-r-rivers t)
;   (s-a-rivers nil) -->
;     (modify 3 ^2 sa-streams)
;     )
;;
;;  600 WRITE(LU(2),605)
(setq rega-sa-drg-msg-1
      '(t t 5 "We are now interested in the fresh water rivers" t
        5 "and streams of the study area." t
        5 "Are there any rivers in the study area?" t t))
;;      READ(LU(1),*)ISAC
;;      IF(ISAC.NE.1)GO TO 640
;;
;(p rega-sa-meandering-river-rule
;   (goal rega)
;   (rega drainage)
;   (drainage sa-rivers)
;   (s-a-rivers t)
;   (s-a-meandering-rivers unasked) -->
;     (call display rega-sa-drg-msg-2)
```

```
;       (call yes-no s-a-meandering-rivers)
;       (remove 5)
;       )
;;
;;        WRITE(LU(2),610)
(setq rega-sa-drg-msg-2
       '(t t 5 "Does any river in the study area have" t
         5 "meanders?" t t))
;;        READ(LU(1),*)ISACM
;;        IF(ISACM.NE.1)GO TO 620
;;
;;        IF(MAXS.LT.100)GO TO 620
;;        WRITE(LU(2),615)
;;
;(p rega-sa-high-meandering-river-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-rivers)
;    (s-a-elevation {} > 99)
;    (s-a-meandering-rivers t)
;    (s-a-high-meandering-rivers unasked) -->
;      (call display rega-sa-drg-msg-3)
;      (call yes-no s-a-high-meandering-rivers)
;      (remove 6)
;      )
(setq rega-sa-drg-msg-3
       '(t t 5 "Do any of the river meanders" t
         5 "occur above the elevation of 100'?" t t))
;;        READ(LU(1),*)ISACMH
;;
;(p rega-sa-rapid-fall-river-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-rivers)
;    (s-a-rivers t)
;    (s-a-rapid/fall-rivers unasked) -->
;      (call display rega-sa-drg-msg-4)
;      (call yes-no s-a-rapid/fall-rivers)
;      (remove 5)
;      )
;;
;;  620 WRITE(LU(2),625)
(setq rega-sa-drg-msg-4
       '(t t 5 "Does the map indicate the existance of" t
         5 "rapids or falls in the study area rivers?" t t))
;;        READ(LU(1),*)ISARF
;;
;(p rega-sa-deflected-river-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-rivers)
;    (s-a-rivers t)
;    (s-a-rapid/fall-rivers <> unasked)
;    (s-a-deflected-rivers unasked) -->
;      (call display rega-sa-drg-msg-5)
;      (call yes-no s-a-deflected-rivers)
;      (remove 6)
;      )
;;
```

```
;;       WRITE(LU(2),630)
(setq rega-sa-drg-msg-5
      '(t t 5 "Does any river in the study area" t
        5 "have a sharp alignment change along the channel" t
        5 "centerline?" t t))
;;       READ(LU(1),*)ISACD
;;       IF(ISACD.NE.1)GO TO 640
;;
;(p rega-sa-high-deflected-river-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-rivers)
;    (s-a-rivers t)
;    (s-a-elevation {> > 99)
;    (s-a-deflected-rivers t)
;    (s-a-high-deflected-rivers unasked) -->
;      (call display rega-sa-drg-msg-6)
;      (call yes-no s-a-high-deflected-rivers)
;      (remove 7)
;      )
;;
;;       IF(MAXS.LT.100)GO TO 640
;;       WRITE(LU(2),635)
(setq rega-sa-drg-msg-6
      '(t t 5 "Do any of the sharp alignment changes" t
        5 "occur above the elevation of 100'?" t t))
;;       READ(LU(1),*)ISACDH
;;
;(p rega-sa-river-to-stream-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-rivers)
;    (s-a-rivers t) -->
;      (modify 3 ^2 sa-streams)
;      (remove 3)
;      )
;;
;;   640 WRITE(LU(2),645)
;;
;(p rega-sa- ream-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-streams)
;    (s-r-streams t)
;    (s-a-streams unasked) -->
;      (call display rega-sa-drg-msg-7)
;      (call yes-no s-a-streams)
;      (remove 5)
;      )
;;
;(p rega-sa-no-sr-stream-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-streams)
;    (s-r-streams nil)
;    (s-a-streams unasked) -->
;      (modify 2 ^2 special)
;      (remove 3)
;      (modify 5 ^2 nil rega-sa-no-sr-stream-rule)
```

```
;       )
;;
;(p rega-sa-no-sa-stream-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-streams)
;    (s-r-streams t)
;    (s-a-streams nil) -->
;      (modify 2 ^2 special)
;      (remove 3)
;      )
;;
(setq rega-sa-drg-msg-7
      '(t t 5 "Do any streams exist in the" t
        5 "study area?" t t))
;;      READ(LU(1),*)ISS
;;      IF(ISS.NE.1)GO TO 700
;;
;(p rega-sa-drg-meander-str-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-streams)
;    (s-a-streams t)
;    (s-a-meandering-streams unasked) -->
;      (call display rega-sa-drg-msg-8)
;      (call yes-no s-a-meandering-streams)
;      (remove 5)
;      )
;;
;;      WRITE(LU(2),650)
(setq rega-sa-drg-msg-8
      '(t t 5 "Are there any meander segments" t
        5 "along the streams in the study area?" t t))
;;      READ(LU(1),*)ISSM
;;      IF(ISSM.NE.1)GO TO 660
;;
;(p rega-sa-high-meander-stream-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-streams)
;    (s-a-elevation {} > 99)
;    (s-a-meandering-streams t)
;    (s-a-high-meandering-streams unasked) -->
;      (call display rega-sa-drg-msg-9)
;      (call yes-no s-a-high-meandering-streams)
;      (remove 6)
;      )
;;
;;      IF(MAXS.LT.100)GO TO 660
;;      WRITE(LU(2),655)
(setq rega-sa-drg-msg-9
      '(t t 5 "Do any of the meander segments occur" t
        5 "above the elevation of 100'?" t t))
;;      READ(LU(1),*)ISSMH
;;
;(p rega-sa-dseg-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-streams)
```

```
;    (s-a-meandering-streams <> unasked)
;    (s-a-distributary-segments unasked) -->
;      (call display rega-sa-drg-msg-10)
;      (call yes-no s-a-distributary-segments)
;      (remove 5)
;      )
;;
;;   660 WRITE(LU(2),665)
(setq rega-sa-drg-msg-10
      '(t t 5 "Do any of the streams have" t
        5 "distributary segments?" t t))
;;      READ(LU(1),*)ISSD
;;      IF(ISSD.NE.1)GO TO 675
;;
;(p rega-sa-high-dseg-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-streams)
;    (s-a-elevation <> > 99)
;    (s-a-distributary-segments t)
;    (s-a-high-distributary-segments unasked) -->
;      (call display rega-sa-drg-msg-11)
;      (call yes-no s-a-high-distributary-segments)
;      (remove 6)
;      )
;;
;;      IF(MAXS.LT.100)GO TO 675
;;      WRITE(LU(2),670)
(setq rega-sa-drg-msg-11
      '(t t 5 "Do any of these distributary channels" t
        5 "occur above the elevation of 100'?" t t))
;;      READ(LU(1),*)ISSDH
;;
;(p rega-sa-gradient-stream-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-streams)
;    (s-a-distributary-segments <> unasked)
;    (s-a-strange-gradients unasked) -->
;      (call display rega-sa-drg-msg-12)
;      (call yes-no s-a-strange-gradients)
;      (remove 5)
;      )
;;
;;   675 WRITE(LU(2),680)
(setq rega-sa-drg-msg-12
      '(t t 5 "Now scan the streams in the study" t
        5 "from low elevations to high elevations to detect" t
        5 "unexpected gradient changes. Are there any unexpected" t
        5 "gradient changes?" t t))
;;      READ(LU(1),*)ISSG
;;      IF(ISSG.NE.1)GO TO 700
;;
;(p rega-sa-gradient-incised-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-streams)
;    (s-a-strange-gradients t)
;    (s-a-strange-gradients-incised unasked) -->
```

```
;       (call display rega-sa-drg-msg-13)
;       (call yes-no s-a-strange-gradients-incised)
;       (remove 5)
;       )
;;
;;        WRITE(LU(2),685)
(setq rega-sa-drg-msg-13
       '(t t 5 "Is the stream incised and" t
         5 "the valley walls constricted in the vicinity of the" t
         5 "gradient change?" t t))
;;        READ(LU(1),*)ISSGI
;;
;(p rega-sa-str-done-rule
;    (goal rega)
;    (rega drainage)
;    (drainage sa-streams) -->
;      (modify 2 ^2 special)
;      (remove 3)
;      )
;;
;; ************************************************************************
;;  SPECIAL QUESTIONS ABOUT STUDY REGION AND STUDY AREA
;; ************************************************************************
;;
;(p rega-special-rule
;    (goal rega)
;    (rega special)
;    (s-r-gravel-pits unasked)
;    (s-a-gravel-pits unasked)
;    (s-r-quarry unasked)
;    (s-a-quarry unasked) -->
;      (call display rega-special-msg)
;      (call display rega-spc-pits-msg-1)
;      (call pause)
;      (call display rega-spc-pits-msg-2)
;      (remove 3)
;      (make s-r-gravel-pits (acceptline))
;      (call display rega-spc-pits-msg-3)
;      (remove 4)
;      (make s-a-gravel-pits (acceptline))
;      (call display rega-spc-quarry-msg-1)
;      (remove 5)
;      (make s-r-quarry (acceptline))
;      (call display rega-spc-quarry-msg-2)
;      (remove 6)
;      (make s-a-quarry (acceptline))
;      (call clear)
;      (modify 2 ^2 listing)
;      )
;;
;;   700 WRITE(LU(2),705)
(setq rega-special-msg
       '(t t 5 "We now want to consider detection of selected" t
         5 "features in the study region and study area which" t
         5 "have significance for recognition and detection" t
         5 "of landform types." t t))
;;        WRITE(LU(2),710)
(setq rega-spc-pits-msg-1
       '(t t 5 "Search the study region outside the study" t
```

```
        5 "area to locate gravel pits and note their" t
        5 "associated surface elevations." t t))
;;      PAUSE 11
;;
;;      WRITE(LU(2),720)
(setq rega-spc-pits-msg-2
      '(t t 5 "If you have located any gravel pits in the study" t
        5 "region, then please type in their surface elevations" t
        5 "on a line separated by spaces and surrounded by ()s," t
        5 "else just type ()<cr>." t t))
;;      READ(LU(1),*)IRGP(1)
;;      IF(IRGP(1).EQ.0)GO TO 735
;;      DO 730 I=2,20
;;      WRITE(LU(2),725)
;;(setq rega-msg
;;      '(t t 5 "If you have found another gravel pit in the" t
;;        5 "study region type its surface elevation, otherwise" t
;;        5 "type '0'." t t))
;;      READ(LU(1),*)IRGP(I)
;;      IF(IRGP(I).EQ.0)GO TO 735
;;  730 CONTINUE
;;
;;  735 WRITE(LU(2),740)
(setq rega-spc-pits-msg-3
      '(t t 5 "Now search the study area for gravel pits." t
        5 "Type a line containing the elevations of all the" t
        5 "gravel pits found in the study area." t
        5 "Remember the ()s." t t))
;;      READ(LU(1),*)ISGP(1)
;;      IF(ISGP(1).EQ.0)GO TO 755
;;      DO 750 I=2,20
;;      WRITE(LU(2),745)
;;(setq rega-msg
;;      '(t t 5 "If you have located another gravel pit in" t
;;        5 "the study area type its surface elevation, if not" t
;;        5 "type '0'." t t))
;;      READ(LU(1),*)ISGP(I)
;;      IF(ISGP(I).EQ.0)GO TO 755
;;  750 CONTINUE
;;
;;  755 WRITE(LU(2),760)
(setq rega-spc-quarry-msg-1
      '(t t 5 "Now search the study region for quarries." t
        5 "Type a line containg the elevations of all the" t
        5 "quarries found in the study region." t
        5 "Remember the ()s." t t))
;;      READ(LU(1),*)IRRQ(1)
;;      IF(IRRQ(1).EQ.0)GO TO 775
;;      DO 770 I=2,20
;;      WRITE(LU(2),765)
;;(setq rega-msg
;;      '(t t 5 "If you find another quarry in the study region" t
;;        5 "type its surface elevation, otherwise type '0'." t t))
;;      READ(LU(1),*)IRRQ(I)
;;      IF(IRRQ(I).EQ.0)GO TO 775
;;  770 CONTINUE
;;
;;  775 WRITE(LU(2),776)
(setq rega-spc-quarry-msg-2
```

```
       '(t t 5 "Now search the study area for quarries." t
         5 "Type a line containing the elevations of all the" t
         5 "quarries found in the study area." t
         5 "Remember the ()s." t t))
;;       READ(LU(1),*)ISRQ(1)
;;       IF(ISRQ(1).EQ.0)GO TO 795
;;       DO 790 I=2,20
;;       WRITE(LU(2),780)
;;(setq rega-msg
;;       '(t t 5 "If you find another quarry in the study area," t
;;         5 "type its surface elevation, otherwise type '0'." t t))
;;       READ(LU(1),*)ISRQ(I)
;;       IF(ISRQ(I).EQ.0)GO TO 795
;;  790 CONTINUE
;;
;;  795 CONTINUE
;;
;; ************************************************************************
;;  REGIONAL ANALYSIS SUMMARY
;; ************************************************************************
;;
;; 1000 CONTINUE
;;
;; ------------------------------------------------------------------------
;;  STATUS OF REGIONAL ANALYSIS DECISIONS AT END OF PROGRAM
;; ------------------------------------------------------------------------
;;
;(p rega-list-rule-1
;    (goal rega)
;    (rega listing)
;    (s-r-elevation <srmin> <srmax>)
;    (s-a-elevation <samin> <samax>)
;    (s-r-coastal-shoreline <irc>)
;    (s-r-coastal-shoreline-ext <irce>)
;    (s-r-tidal-river <irtr>)
;    (s-r-tidal-river-ext <irtre>)
;    (s-r-tidal-river-islands <irtri>)
;    (s-r-low-coastal-shoreline <irlc>)
;    (s-r-low-tidal-river <irltr>)
;    (s-r-low-alluvial-channel <irlac>)
;    (s-r-tidal-river-rock <irtrr>)
;    (s-r-tidal-river-mud <irtrm>)
;    (s-r-tidal-river-sand <irtrs>)
;    (s-r-low-terraces <irtlo>)
;    (s-r-low-terraces-near-coast <irtlc>)
;    (s-r-low-terraces-near-tidal-river <irtltr>)
;    (s-r-mid-terraces <irt2o>)
;    (s-r-high-terraces <irt3o>)
;    (s-r-rivers <irac>)
;    (s-r-meandering-rivers <iracm>)
;    (s-r-low-meandering-rivers <iracml>)
;    (s-r-high-meandering-rivers <iracmh>)
;    (s-r-rapid/fall-rivers <irarf>)
;    (s-r-deflected-rivers <iracd>)
;    (s-r-high-deflected-rivers <iracdh>)
;    (s-r-streams <irs>)
;    (s-r-meandering-streams <irsm>)
;    (s-r-high-meandering-streams <irsmh>)
;    (s-r-distributary-segments <irsd>)
```

```
;    (s-r-high-distributary-segments <irsdh>)
;    (s-r-strange-gradients <irsg>)
;    (s-r-strange-gradients-incised <irsgi>)
;    (s-r-gravel-pits <irgp1> <irgp2> <irgp3> <irgp4> <irgp5>)
;    (s-r-quarry <irrq1> <irrq2> <irrq3> <irrq4> <irrq5>)
;    (s-a-coastal-shoreline <isc>)
;    (s-a-coastal-shoreline-ext <isce>)
;    (s-a-tidal-river <istr>)
;    (s-a-tidal-river-ext <istre>)
;    (s-a-tidal-river-islands <istri>)
;    (s-a-low-coastal-shoreline <islc>)
;    (s-a-low-tidal-river <isltr>)
;    (s-a-low-alluvial-channel <islac>)
;    (s-a-max-coastal-shoreline <ismc>)
;    (s-a-max-tidal-river <ismtr>)
;    (s-a-max-un-associated <ismz>)
;    (s-a-tidal-river-rock <istrr>)
;    (s-a-tidal-river-mud <istrm>)
;    (s-a-tidal-river-sand <istrs>)
;    (s-a-low-terraces <istlo>)
;    (s-a-low-terraces-near-coast <istlc>)
;    (s-a-low-terraces-near-tidal-river <istltr>)
;    (s-a-mid-terraces <ist2o>)
;    (s-a-high-terraces <ist3o>)
;    (s-a-rivers <isac>)
;    (s-a-meandering-rivers <isacm>)
;    (s-a-high-meandering-rivers <isacmh>)
;    (s-a-rapid/fall-rivers <isarf>)
;    (s-a-deflected-rivers <isacd>)
;    (s-a-high-deflected-rivers <isacdh>)
;    (s-a-streams <iss>)
;    (s-a-meandering-streams <issm>)
;    (s-a-high-meandering-streams <issmh>)
;    (s-a-distributary-segments <issd>)
;    (s-a-high-distributary-segments <issdh>)
;    (s-a-strange-gradients <issg>)
;    (s-a-strange-gradients-incised <issgi>)
;    (s-a-gravel-pits <isgp1> <isgp2> <isgp3> <isgp4> <isgp5>)
;    (s-a-quarry <isrq1> <isrq2> <isrq3> <isrq4> <isrq5>)
;    -->
;      (call display rega-listing-start-msg)
;      (call pause)
;      (call display rega-list-header-msg)
;      (call display rega-list-elv-msg <srmin> <srmax> <samin> <samax>)
;      (call display rega-list-sr-msg-1)
;      (call display-tf rega-list-sr-msg-2 <irc> <irce> <irtr> <irtre>)
;      (call display-tf rega-list-sr-msg-3 <irtri> <irlc> <irltr> <irlac>)
;      (call display-tf rega-list-sr-msg-4 <irtrr> <irtrm> <irtrs> <irtlo>)
;      (call display-tf rega-list-sr-msg-5 <irtlc> <irtltr> <irt2o> <irt3o>)
;      (call display-tf rega-list-sr-msg-6 <irac> <iracm> <iracml>)
;      (call display-tf rega-list-sr-msg-7 <iracmh> <irarf> <iracd>)
;      (call display-tf rega-list-sr-msg-8 <iracdh> <irs> <irsm> <irsmh>)
;      (call display-tf rega-list-sr-msg-9 <irsd> <irsdh> <irsg> <irsgi>)
;      (call display-and rega-list-sr-msg-10
;                        <irgp1> <irgp2> <irgp3> <irgp4> <irgp5>)
;      (call display-and rega-list-sr-msg-11
;                        <irrq1> <irrq2> <irrq3> <irrq4> <irrq5>)
;      (call display rega-list-sa-msg-1)
;      (call display-tf rega-list-sa-msg-2 <isc> <isce> <istr> <istre>)
```

```
;       (call display-tf rega-list-sa-msg-3 <istri> <islc> <isltr> <islac>)
;       (call display-tf rega-list-sa-msg-4 <ismc> <ismtr> <ismz> <istrr>)
;       (call display-tf rega-list-sa-msg-5 <istrm> <istrs> <ist1o> <ist1c>)
;       (call display-tf rega-list-sa-msg-6 <ist1tr> <ist2o> <ist3o> <isac>)
;       (call display-tf rega-list-sa-msg-7 <isacm> <isacmh> <isarf> <isacd>)
;       (call display-tf rega-list-sa-msg-8 <isacdh> <iss> <issm> <issmh>)
;       (call display-tf rega-list-sa-msg-9 <issd> <issdh> <issg> <issgi>)
;       (call display-and rega-list-sa-msg-10
;                         <isgp1> <isgp2> <isgp3> <isgp4> <isgp5>)
;       (call display-and rega-list-sa-msg-11
;                         <isrq1> <isrq2> <isrq3> <isrq4> <isrq5>)
;       )
;;
;; 1500 WRITE(LU(2),1502)
(setq rega-listing-start-msg
      '(t t "I'll wait till you are ready to list the"
        t " REGIONAL ANALYSIS SUMMARY" t t))
;;      PAUSE 12
;;      WRITE(LU(3),1505)
(setq rega-list-header-msg
      '(t t t 5 "STATUS OF REGIONAL ANALYSIS DECISIONS" t
        5 "          AT END OF PROGRAM" t t))
;;
;;      WRITE(LU(3),1510)MINR,MAXR,MINS,MAXS
(setq rega-list-elv-msg
      '("STUDY REGION:  min= " (o 1) ",  max= " (o 2) t
        "STUDY AREA:    min= " (o 3) ",  max= " (o 4) t t))
;;
;;      WRITE(LU(3),1515)
(setq rega-list-sr-msg-1
      '(t t 5 "STUDY REGION DECISIONS:" t))
;;      WRITE(LU(3),1520)IRC,IRCE,IRTR,IRTRE,IRTRI,IRLC,IRLTR,IRLAC,
;;        IRTRR,IRTRM,IRTRS,IRT1O,IRT1C,IRT1TR,IRT2O,IRT3O,IRAC,IRACM,
;;        IRACML,IRACMH,IRARF,IRACD,IRACDH,IRS,IRSM,IRSMH,IRSD,
;;        IRSDH,IRSG,IRSGI
(setq rega-list-sr-msg-2
      '(t " IRC = " (o 1) ",  IRCE = " (o 2)
        ",  IRTR = " (o 3) ",  IRTRE = " (o 4)))
(setq rega-list-sr-msg-3
      '(t " IRTRI = " (o 1) ",  IRLC = " (o 2)
        ",  IRLTR = " (o 3) ",  IRLAC = " (o 4)))
(setq rega-list-sr-msg-4
      '(t " IRTRR = " (o 1) ",  IRTRM = " (o 2)
        ",  IRTRS = " (o 3) ",  IRT1O = " (o 4)))
(setq rega-list-sr-msg-5
      '(t " IRT1C = " (o 1) ",  IRT1TR = " (o 2)
        ",  IRT2O = " (o 3) ",  IRT3O = " (o 4)))
(setq rega-list-sr-msg-6
      '(t " IRAC = " (o 1) ",  IRACM = " (o 2) ",  IRACML = " (o 3)))
(setq rega-list-sr-msg-7
      '(t " IRACMH = " (o 1) ",  IRARF = " (o 2) ",  IRACD = " (o 3)))
(setq rega-list-sr-msg-8
      '(t " IRACDH = " (o 1) ",  IRS = " (o 2)
        ",  IRSM = " (o 3) ",  IRSMH = " (o 4)))
(setq rega-list-sr-msg-9
      '(t " IRSD = " (o 1) ",  IRSDH = " (o 2)
        ",  IRSG = " (o 3) ",  IRSGI = " (o 4) t))
;;
;;      IF(IRGP(1).EQ.0)GO TO 1524
```

```
;;       WRITE(LU(3),1521)
(setq rega-list-sr-msg-10
      '(t t 5 "ELEVATION OF GRAVEL PITS IN STUDY REGION" t
        10 (o 1) " " (o 2) " " (o 3) " " (o 4) " " (o 5)))
;;       DO 1523 I=1,20
;;       IF(IRGP(I).EQ.0)GO TO 1524
;;       WRITE(LU(3),1522)I,IRGP(I)
;; 1523 CONTINUE
;;
;; 1524 IF(IRRG(1).EQ.0)GO TO 1550
;;       WRITE(LU(3),1525)
(setq rega-list-sr-msg-11
      '(t t 5 "ELEVATIONS OF QUARRIES IN STUDY REGION:" t
        10 (o 1) " " (o 2) " " (o 3) " " (o 4) " " (o 5)))
;;       DO 1527 I=1,20
;;       IF(IRRG(I).EQ.0)GO TO 1550
;;       WRITE(LU(3),1522)I,IRRG(I)
;; 1527 CONTINUE
;;
;; ---------------------------------------------------------------
;;  STUDY AREA DECISIONS
;; ---------------------------------------------------------------
;;
;; 1550 WRITE(LU(3),1555)
(setq rega-list-sa-msg-1
      '(t t 5 "STUDY AREA DECISIONS:" t))
;;       WRITE(LU(3),1560)ISC,ISCE,ISTR,ISTRE,ISTRI,ISLC,ISLTR,ISLAC,
;;         ISMC,ISMTR,ISMZ,ISTRR,ISTRM,ISTRS,IST10,IST1C,IST1TR,IST20,
;;         IST30,ISAC,ISACM,ISACMH,ISARF,ISACD,ISACDH,ISS,ISSM,ISSMH,
;;         ISSD,ISSDH,ISSQ,ISSQI
(setq rega-list-sa-msg-2
      '(t " ISC = " (o 1) ", ISCE = " (o 2)
        ", ISTR = " (o 3) ", ISTRE = " (o 4)))
(setq rega-list-sa-msg-3
      '(t " ISTRI = " (o 1) ", ISLC = " (o 2)
        ", ISLTR = " (o 3) ", ISLAC = " (o 4)))
(setq rega-list-sa-msg-4
      '(t " ISMC = " (o 1) ", ISMTR = " (o 2)
        ", ISMZ = " (o 3) ", ISTRR = " (o 4)))
(setq rega-list-sa-msg-5
      '(t " ISTRM = " (o 1) ", ISTRS = " (o 2)
        ", IST10 = " (o 3) ", IST1C = " (o 4)))
(setq rega-list-sa-msg-6
      '(t " IST1TR = " (o 1) ", IST20 = " (o 2)
        ", IST30 = " (o 3) ", ISAC = " (o 4)))
(setq rega-list-sa-msg-7
      '(t " ISACM = " (o 1) ", ISACMH = " (o 2)
        ", ISARF = " (o 3) ", ISACD = " (o 4)))
(setq rega-list-sa-msg-8
      '(t " ISACDH = " (o 1) ", ISS = " (o 2)
        ", ISSM = " (o 3) ", ISSMH = " (o 4)))
(setq rega-list-sa-msg-9
      '(t " ISSD = " (o 1) ", ISSDH = " (o 2)
        ", ISSQ = " (o 3) ", ISSQI = " (o 4)))
;;
;;       IF(ISQP(1).EQ.0)GO TO 1580
;;       WRITE(LU(3),1565)
(setq rega-list-sa-msg-10
      '(t t 5 "ELEVATIONS OF GRAVEL PITS IN STUDY AREA:" t
```

```
        10 (o 1) " " (o 2) " " (o 3) " " (o 4) " " (o 5)))
;;      DO 1575 I=1,20
;;      IF(ISGP(I).EQ.0)GO TO 1580
;;      WRITE(LU(3),1522)I,ISGP(I)
;;1575 CONTINUE
;;
;;1580 IF(ISRG(1).EQ.0)GO TO 1600
;;      WRITE(LU(3),1585)
(setq rega-list-sa-msg-11
      '(t t 5 "ELEVATIONS OF ROCK QUARRIES IN STUDY AREA:" t
        10 (o 1) " " (o 2) " " (o 3) " " (o 4) " " (o 5)))
;;      DO 1590 I=1,20
;;      IF(ISRG(I).EQ.0)GO TO 1600
;;      WRITE(LU(3),1522)I,ISRG(I)
;;1590 CONTINUE
;;
;;1600 CONTINUE
;;
;(p rega-listing-done-rule
;   (goal rega)
;   (rega listing) -->
;     (modify 2 ^2 done)
;     )
;;
;;*****************************************************************************
;;
;;  RETURN TO EXECUTIVE PROGRAM SEGMENT
;;   TO BEGIN DETAILED STUDY AREA ANALYSIS
;;
;;*****************************************************************************
;;      CALL EXEC(8,ICALAP)
;;
;;      END
(terpr)
(princ "REGA.L Loaded")
(terpr)
```

REGIN.L

```
;
;       regin.l     Regional Initialization Program
;
;       Fortran System:   Robert Leighty    USAETL
;       OPS5 System:      Kenneth Hayes     Smart Systems Technology
;       DUCK System:      Kenneth Hayes     Smart Systems Technology
;
;;      SUBROUTINE REGIN
;;
;; ****************************************************************
;;
;;  INITIALIZES VARIABLES IN SEGMENTED PROGRAM REGA
;;           (SEE GLOSSARY IN FILE 'REGGLO')
;;
;;  RDL  MAY 1981
;; ****************************************************************
;;
;;      COMMON LU(5),MLAT1(3),MLON1(3),MLAT2(3),MLON2(3),ISTEP,MINR,
;;     *  MAXR,MINS,MAXS
;;
;;      COMMON  IRC,ISC,IRCE,ISCE,IRTR,ISTR,IRTRE,ISTRE,IRLC,
;;     *  IRLAC,ISLC,ISLTR,ISLAC,ISMC,ISMTR,IRTRR,IRTRM,IRTRS,ISTRR,
;;     *  ISTRM,ISTRS,ISMZ,IRT1O,IRT1C,IRT1TR,IST1O,IST1C,IST1TR,IRT2O,
;;     *  IST2O,IRT3O,IST3O,IRAC,IRACM,IRACML,IRACMH,IRARF,IRACD,IRACDH,
;;     *  IRS,IRSM,IRSMH,IRSD,IRSDH,IRSG,IRSGI,ISAC,ISACM,ISACMH,ISARF,
;;     *  ISACD,ISACDH,ISS,ISSM,ISSDH,ISSG,ISSGI,IRLTR,ISSD,ISSMH,
;;     *  IRGP(20),IRRG(20),ISGP(20),ISRQ(20),IRTRI,ISTRI
;;
;;  ---------------------------------------------------------------
;;  INITIALIZE STUDY REGION DECISION VARIABLES
;;  ---------------------------------------------------------------
;;
;
(lisprule regin-unasked-rule (feature ?x) ->
  (neglecting '(ansed ?x)
    (premiss '(unasked ?x))
    )
  )
;
(rule answered-is-ansed
  (-> (answer ?x ?y) (ansed ?x))
  )
;
(lisprule regin-rule (goal regin) ->
  (for (x in (get 'FEATURE 'examples))
       (do (premiss `(feature ,x)))
       )
  (for (x in '(s-r-gravel-pits s-r-quarry s-a-gravel-pits s-a-quarry))
       (do (premiss `(spfeature ,x)))
       )
  (premiss '(done regin))
  )
;
;(p regin-sr-rule
;   (goal regin) -->
;     (make s-r-coastal-shoreline              unasked)
;     (make s-r-coastal-shoreline-ext          unasked)
;     (make s-r-tidal-river                    unasked)
;     (make s-r-tidal-river-ext                unasked)
```

```
;       (make s-r-tidal-river-islands              unasked)
;       (make s-r-low-coastal-shoreline            unasked)
;       (make s-r-low-tidal-river                  unasked)
;       (make s-r-low-alluvial-channel             unasked)
;       (make s-r-tidal-river-rock                 unasked)
;       (make s-r-tidal-river-mud                  unasked)
;       (make s-r-tidal-river-sand                 unasked)
;       (make s-r-low-terraces                     unasked)
;       (make s-r-low-terraces-near-coast          unasked)
;       (make s-r-low-terraces-near-tidal-river    unasked)
;       (make s-r-mid-terraces                     unasked)
;       (make s-r-high-terraces                    unasked)
;       (make s-r-rivers                           unasked)
;       (make s-r-meandering-rivers                unasked)
;       (make s-r-low-meandering-rivers            unasked)
;       (make s-r-high-meandering-rivers           unasked)
;       (make s-r-rapid/fall-rivers                unasked)
;       (make s-r-deflected-rivers                 unasked)
;       (make s-r-high-deflected-rivers            unasked)
;       (make s-r-streams                          unasked)
;       (make s-r-meandering-streams               unasked)
;       (make s-r-high-meandering-streams          unasked)
;       (make s-r-distributary-segments            unasked)
;       (make s-r-high-distributary-segments       unasked)
;       (make s-r-strange-gradients                unasked)
;       (make s-r-strange-gradients-incised        unasked)
;       (make s-r-init)
;       )
;;
;;        IRC=2
;;        IRCE=2
;;        IRTR=2
;;        IRTRE=2
;;        IRTRI=2
;;        IRLC=2
;;        IRLTR=2
;;        IRLAC=2
;;        IRTRR=2
;;        IRTRM=2
;;        IRTRS=2
;;        IRT1O=2
;;        IRT1C=2
;;        IRT1TR=2
;;        IRT2O=2
;;        IRT3O=2
;;        IRAC=2
;;        IRACM=2
;;        IRACML=2
;;        IRACMH=2
;;        IRARF=2
;;        IRACD=2
;;        IRACDH=2
;;        IRS=2
;;        IRSM=2
;;        IRSMH=2
;;        IRSD=2
;;        IRSDH=2
;;        IRSG=2
;;        IRSGI=2
```

```
;;
;; ------------------------------------------------------------
;;  INITIALIZATION OF STUDY AREA DECISIONS
;; ------------------------------------------------------------
;;
;(p regin-sa-rule
;   (goal regin) -->
;     (make s-a-coastal-shoreline                unasked)
;     (make s-a-coastal-shoreline-ext            unasked)
;     (make s-a-tidal-river                      unasked)
;     (make s-a-tidal-river-ext                  unasked)
;     (make s-a-tidal-river-islands              unasked)
;     (make s-a-low-coastal-shoreline            unasked)
;     (make s-a-low-tidal-river                  unasked)
;     (make s-a-low-alluvial-channel             unasked)
;     (make s-a-max-coastal-shoreline            unasked)
;     (make s-a-max-tidal-river                  unasked)
;     (make s-a-max-un-associated                unasked)
;     (make s-a-tidal-river-rock                 unasked)
;     (make s-a-tidal-river-mud                  unasked)
;     (make s-a-tidal-river-sand                 unasked)
;     (make s-a-low-terraces                     unasked)
;     (make s-a-low-terraces-near-coast          unasked)
;     (make s-a-low-terraces-near-tidal-river    unasked)
;     (make s-a-mid-terraces                     unasked)
;     (make s-a-high-terraces                    unasked)
;     (make s-a-rivers                           unasked)
;     (make s-a-meandering-rivers                unasked)
;     (make s-a-low-meandering-rivers            unasked)
;     (make s-a-high-meandering-rivers           unasked)
;     (make s-a-rapid/fall-rivers                unasked)
;     (make s-a-deflected-rivers                 unasked)
;     (make s-a-high-deflected-rivers            unasked)
;     (make s-a-streams                          unasked)
;     (make s-a-meandering-streams               unasked)
;     (make s-a-high-meandering-streams          unasked)
;     (make s-a-distributary-segments            unasked)
;     (make s-a-high-distributary-segments       unasked)
;     (make s-a-strange-gradients                unasked)
;     (make s-a-strange-gradients-incised        unasked)
;     (make s-a-init)
;     )
;;
;;       ISC=2
;;       ISCE=2
;;       ISTR=2
;;       ISTRE=2
;;       ISTRI=2
;;       ISLC=2
;;       ISLTR=2
;;       ISLAC=2
;;       ISMC=2
;;       ISMTR=2
;;       ISMZ=2
;;       ISTRR=2
;;       ISTRM=2
;;       ISTRS=2
;;       IST1O=2
;;       IST1C=2
```

```
;;        IST1TR=2
;;        IST2O=2
;;        IST3O=2
;;        ISAC=2
;;        ISACM=2
;;        ISCAMH=2
;;        ISARF=2
;;        ISACD=2
;;        ISACDH=2
;;        ISS=2
;;        ISSM=2
;;        ISSMH=2
;;        ISSD=2
;;        ISSDH=2
;;        ISSG=2
;;        ISSGI=2
;;
;(p regin-sr-extra-rule
;    (goal regin) -->
;      (make s-r-unknown -omni-empty-)
;      (make s-r-gravel-pits unasked nil nil nil nil)
;      (make s-r-quarry      unasked nil nil nil nil)
;      (make s-r-e-init)
;      )
;;
;(p regin-sa-extra-rule
;    (goal regin) -->
;      (make s-a-unknown -omni-empty-)
;      (make s-a-coastal-beach               unasked)
;      (make s-a-coastal-beach-ridge         unasked)
;      (make s-a-coastal-swale               unasked)
;      (make s-a-coastal-sand                unasked)
;      (make s-a-coastal-ponded-depression unasked)
;      (make s-a-coastal-islands             unasked)
;      (make s-a-coastal-bars                unasked)
;      (make s-a-coastal-bed-rock            unasked)
;      (make s-a-gravel-pits                 unasked nil nil nil nil)
;      (make s-a-quarry                      unasked nil nil nil nil)
;      (make s-a-e-init)
;      )
;;
;;        RETURN
;;
;(p regin-done-rule
;    (goal regin)
;    (s-r-init)
;    (s-a-init)
;    (s-r-e-init)
;    (s-a-e-init) -->
;      (modify 1 ^1 done)
;      (remove 2 3 4 5)
;      )
;;
;;        END
(terpr)
(princ "REGIN.L Loaded")
(terpr)
```

REGAS.L

```
;
;       regas.l     Regional Analysis Summary Program
;
;       Fortran System:   Robert Leighty    USAETL
;       OPS5 System:      Kenneth Hayes     Smart Systems Technology
;       DUCK System:      Kenneth Hayes     Smart Systems Technology
;
;;      PROGRAM REGAS(5)
;;
;;  ***************************************************************
;;   REGIONAL ANALYSIS SUMMARY SEGMENT OF PROGRAM CALAP.
;;
;;   RDL, MAY 81
;;  ***************************************************************
;;
;;       COMMON LU(5),MLAT1(3),MLON1(3),MLAT2(3),MLON2(3),ISTEP,MINR,
;;      *   MAXR,MINS,MAXS
;;
;;       COMMON   IRC,ISC,IRCE,ISCE,IRTR,ISTR,IRTRE,ISTRE,IRLC,
;;      *   IRLAC,ISLC,ISLTR,ISLAC,ISMC,ISMTR,IRTRR,IRTRM,IRTRS,ISTRR,
;;      *   ISTRM,ISTRS,ISMZ,IRT1O,IRT1C,IRT1TR,IST1O,IST1C,IST1TR,IRT2O,
;;      *   IST2O,IRT3O,IST3O,IRAC,IRACM,IRACML,IRACMH,IRARF,IRACD,IRACDH,
;;      *   IRS,IRSM,IRSMH,IRSD,IRSDH,IRSG,IRSGI,ISAC,ISACM,ISACMH,ISARF,
;;      *   ISACD,ISACDH,ISS,ISSM,ISSDH,ISSG,ISSGI,IRLTR,ISSD,ISSMH,
;;      *   IRGP(20),IRRQ(20),ISGP(20),ISRQ(20),IRTRI,ISTRI
;;
;;       DIMENSION ICALAP(3)
;;
;;       DATA ICALAP/2HXC,2HAP,2HI /
;;
;;
;;  ---------------------------------------------------------------
;;   INTRODUCTION
;;  ---------------------------------------------------------------
;;
;
(lisprule regas-rule (goal regas) ->
  (display 'regas-summary-msg nil)
  (for-first-ans (fetch '(study-region ?s ?p))
    (display 'regas-study-region-msg (list ?s ?p))
    )
  (display 'regas-sr-elevation-msg
           (list (gans srmin) (gans srmax)))
  (display 'regas-sa-elevation-msg
           (list (gans samin) (gans samax)))
  (premiss '(goal regas-coastal))
  (premiss '(goal regas-tidal-river))
  (premiss '(done regas))
  )
;
(lisprule regas-coastal (goal regas-coastal) ->
  (for [x in '(s-r-coastal-shoreline
               s-r-coastal-shoreline-ext
               s-r-low-coastal-shoreline)]
       [y in '(regas-coastal-msg-1
               regas-coastal-msg-2
               regas-coastal-msg-3)]
       (do (for-first-ans (fetch `(answer ,x ?z))
             (cond [?z (display y (list ?z))]))
```

```
                )
              )
            )
          )
;
(lisprule regas-tidal-river (goal regas-tidal-river) ->
  (for [x in '(s-r-tidal-river
               s-r-tidal-river-rock
               s-r-tidal-river-mud
               s-r-tidal-river-sand
               s-r-tidal-river-ext
               s-r-low-tidal-river
               s-r-low-terraces
               s-r-low-terraces-near-coast
               s-r-low-terraces-near-tidal-river)]
       [y in '(regas-tidal-msg-1
               regas-tidal-msg-2
               regas-tidal-msg-3
               regas-tidal-msg-4
               regas-tidal-msg-5
               regas-tidal-msg-6
               regas-tidal-msg-7
               regas-tidal-msg-8
               regas-tidal-msg-9)]
       (do (for-first-ans (fetch '(answer ,x ?z))
             (cond [?z (display y (list ?z))])
             )
           )
       )
  )
;
;(p regas-rule-1
;   (goal regas)
;   (study-region <a> <b>) -->
;     (call display regas-summary-msg)
;     (call display regas-study-region-msg <a> <b>)
;     (make regas start)
;     )
;;
;;       WRITE(LU(3),100)
(setq regas-summary-msg
      '(t t t " ************************************************" t t
        10 "SUMMARY OF STUDY REGION ANALYSIS" t
        10 "------- -- ----- ------ --------" t t t))
;;
;(p regas-elevation-rule
;   (goal regas)
;   (s-r-elevation <min-r> <max-r>)
;   (s-a-elevation <min-a> <max-a>)
;   (regas start) -->
;     (call display regas-sr-elevation-msg <min-r> <max-r>)
;     (call display regas-sa-elevation-msg <min-a> <max-a>)
;     )
;;
;;       WRITE(LU(3),105)
(setq regas-study-region-msg
      '(t t 5 "The study region is located in the " (o 1) t
        5 "of the " (o 2) " Physiographic Province." t))
;;
```

```
;;       WRITE(LU(3),110)MINR,MAXR
(setq regas-sr-elevation-msg
      '(t t 5 "The minimum elevation in the study region is " (o 1) t
        5 "and the maximum elevation is " (o 2) "." t))
(setq regas-sa-elevation-msg
      '(t t 5 "The minimum elevation in the study area is " (o 1) t
        5 "and the maximum elevation is " (o 2) "." t))
;;
;; ---------------------------------------------------------------
;;  COASTAL SHORELINE
;; ---------------------------------------------------------------
;;
;(p regas-coastal-rule
;    (goal regas)
;    (s-r-coastal-shoreline <irc>)
;    (s-r-coastal-shoreline-ext <irce>)
;    (s-r-low-coastal-shoreline <irlc>)
;    (regas start) -->
;      (call asked-display <irc> regas-coastal-msg-1)
;      (call asked-display <irce> regas-coastal-msg-2)
;      (call asked-display <irlc> regas-coastal-msg-3)
;      )
;;
;;       IF(IRC.NE.1)GO TO 125
;;       WRITE(LU(3),115)
(setq regas-coastal-msg-1
      '(t t 5 "The study region contains a coastal shoreline." t))
;;       IF(IRCE.NE.1)GO TO 125
;;       WRITE(LU(3),120)
(setq regas-coastal-msg-2
      '(t t 5 "The coastal features appear to extend into the" t
        5 "study area from the coastline." t))
;;
;;   125 IF(IRLC.NE.1)GO TO 135
;;       WRITE(LU(3),130)
(setq regas-coastal-msg-3
      '(t t 5 "Though a coastal shoreline does not exist in the" t
        5 "study region, it is believed that the minimum regional" t
        5 "elevation is associated with a coastal shoreline that" t
        5 "exists outside the study region." t))
;;
;; ---------------------------------------------------------------
;;  TIDAL RIVER
;; ---------------------------------------------------------------
;;
;(p regas-tidal-river-rule
;    (goal regas)
;    (s-r-tidal-river <irtr>)
;    (s-r-tidal-river-rock <irtrr>)
;    (s-r-tidal-river-mud <irtrm>)
;    (s-r-tidal-river-sand <irtrs>)
;    (s-r-tidal-river-ext <irtre>)
;    (s-r-low-tidal-river <irltr>)
;    (s-r-low-terraces <irtlo>)
;    (s-r-low-terraces-near-coast <irtlc>)
;    (s-r-low-terraces-near-tidal-river <irtltr>)
;    (regas start) -->
;      (call asked-display <irtr>   regas-tidal-msg-1)
;      (call asked-display <irtrr>  regas-tidal-msg-2)
```

```
;       (call asked-display <irtrm>  regas-tidal-msg-3)
;       (call asked-display <irtrs>  regas-tidal-msg-4)
;       (call asked-display <irtre>  regas-tidal-msg-5)
;       (call asked-display <irltr>  regas-tidal-msg-6)
;       (call asked-display <irt1o>  regas-tidal-msg-7)
;       (call asked-display <irt1c>  regas-tidal-msg-8)
;       (call asked-display <irt1tr> regas-tidal-msg-9)
;       )
;;
;;   135 IF(IRTR.NE.1)GO TO 170
;;         WRITE(LU(3),140)
(setq regas-tidal-msg-1
      '(t t 5 "The study region contains a tidal river section."))
;;
;;         IF(IRTRR.NE.1)GO TO 150
;;         WRITE(LU(3),145)
(setq regas-tidal-msg-2
      '(t t 5 "The map indicates rock in the river channel."))
;;
;;   150 IF(IRTRM.NE.1)GO TO 160
;;         WRITE(LU(3),155)
(setq regas-tidal-msg-3
      '(t t 5 "The map indicates mud flats exist in the river bed."))
;;
;;   160 IF(IRTRS.NE.1)GO TO 170
;;         WRITE(LU(3),165)
(setq regas-tidal-msg-4
      '(t t 5 "The map indicates sand bars in the river channel."))
;;
;;   170 IF(IRTRE.NE.1)GO TO 180
;;         WRITE(LU(2),175)
(setq regas-tidal-msg-5
      '(t t 5 "Though a tidal river does not exist in the study" t
        5 "region, tidal river features extend into the study" t
        5 "region."))
;;
;;   180 IF(IRLTR.NE.1)GO TO 190
;;         WRITE(LU(2),185)
(setq regas-tidal-msg-6
      '(t t 5 "The regional low is associated with a tidal river" t
        5 "beyond the study region."))
;;
;;   190 IF(IRT1O.NE.1)GO TO 210
;;         WRITE(LU(2),195)
(setq regas-tidal-msg-7
      '(t t 5 "Level land areas between 20' and 50' are found" t
        5 "in the study region."))
;;
;;         IF(IRT1C.NE.1)GO TO 200
;;         WRITE(LU(2),198)
(setq regas-tidal-msg-8
      '(t t 5 "These surfaces are in the vicinity of the coast."))
;;
;;   200 IF(IRT1TR.NE.1)GO TO 210
;;         WRITE(LU(2),205)
(setq regas-tidal-msg-9
      '(t t 5 "These surfaces are in the vicinity of a tidal" t
        5 "river."))
;;
```

```
;;   210 CONTINUE
;;  1000 CONTINUE
;;
;; ---------------------------------------------------------------
;;   RETURN TO EXECUTIVE PROGRAM
;; ---------------------------------------------------------------
;;
;(p regas-done-rule
;    (goal regas)
;    (regas) -->
;      (remove 2)
;      (modify 1 ^1 done)
;      )
;;
;;       CALL EXEC(8,ICALAP)
;;
;;       END
(terpr)
(princ "REGAS.L Loaded")
(terpr)
```

DETA.L

```
;
;       deta.l      Detailed Analysis Program
;
;       Fortran System:   Robert Leighty    USAETL
;       OPS5 System:      Kenneth Hayes     Smart Systems Technology
;       DUCK System:      Kenneth Hayes     Smart Systems Technology
;
;;      PROGRAM DETA(5)
;;
;; ***************************************************************
;;
;;  DETAILED ANALYSIS SEGEMENT FOR PROGRAM CALAP
;;
;;  RDL, MAY 81
;;
;; ***************************************************************
;;
;;      COMMON LU(5),MLAT1(3),MLON1(3),MLAT2(3),MLON2(3),ISTEP,MINR,
;;     *  MAXR,MINS,MAXS
;;
;;      COMMON  IRC,ISC,IRCE,ISCE,IRTR,ISTR,IRTRE,ISTRE,IRLC,
;;     *  IRLAC,ISLC,ISLTR,ISLAC,ISMC,ISMTR,IRTRR,IRTRM,IRTRS,ISTRR,
;;     *  ISTRM,ISTRS,ISMZ,IRT1O,IRT1C,IRT1TR,IST1O,IST1C,IST1TR,IRT2O,
;;     *  IST2O,IRT3O,IST3O,IRAC,IRACM,IRACML,IRACMH,IRARF,IRACD,IRACDH,
;;     *  IRS,IRSM,IRSMH,IRSD,IRSDH,IRSG,IRSGI,ISAC,ISACM,ISACMH,ISARF,
;;     *  ISACD,ISACDH,ISS,ISSM,ISSDH,ISSG,ISSGI,IRLTR,ISSD,ISSMH,
;;     *  IRGP(20),IRRG(20),ISGP(20),ISRG(20),IRTRI,ISTRI
;;
;;      DIMENSION ICALAP(3),IYES(2),INO(2),IBEGIN(3),INEXT(2),IANSD(3)
;;
;;      DATA ICALAP/2HXC,2HAP,2HI /
;;      DATA IYES/2HYE,2HS /,INO/2HNO,2H  /
;;      DATA IBEGIN/2HBE,2HGI,2HN /
;;      DATA INEXT/2HNE,2HXT/
;;
;; ---------------------------------------------------------------
;;  INTRODUCTION
;; ---------------------------------------------------------------
;;
;
(lisprule deta-rule (goal deta) ->
  (display 'deta-area-msg nil)
  (pause)
  (cond [(gans s-a-coastal-shoreline)
         (display 'deta-shore-msg-1 nil)
         (pause)
         (premiss '(goal shore))]
        [(gans s-r-coastal-shoreline-ext)
         (display 'deta-shore-ext-msg nil)
         (premiss '(goal shore))]
        [(gans s-a-low-coastal-shoreline)
         (display 'deta-shore-hide-msg nil)
         (premiss '(goal shore))]
        [t (display 'deta-shore-hide-msg nil)]
        )
  (cond [(gans s-a-tidal-river) (premiss '(goal tidal-river))])
  (premiss '(done deta))
  )
;
```

```
(lisprule deta-shore (goal shore) ->
  (display 'deta-shore-help-msg nil)
  (cond [(is-yes (dread)) (premiss '(goal shore-menu))])
  (cond [(and (not (gans s-a-coastal-shoreline))
              (not (gans s-a-coastal-shoreline-ext)))
         (display 'deta-coast-in-sr-sa-msg nil)])
  (premiss '(ask s-a-coastal-beach
                 deta-beach-do-msg-1
                 deta-beach-do-msg-2))
  (premiss '(ask s-a-coastal-beach-ridge
                 deta-beach-ridge-do-msg-1
                 deta-beach-ridge-do-msg-2))
  (premiss '(ask s-a-coastal-swale-ridge
                 deta-swale-do-msg-1
                 deta-swale-do-msg-2))
  (premiss '(ask s-a-coastal-sand
                 deta-sand-do-msg-1
                 deta-sand-do-msg-2))
  (premiss '(ask s-a-coastal-ponded-depression
                 deta-ponded-do-msg-1
                 deta-ponded-do-msg-2))
  (premiss '(ask s-a-coastal-islands
                 deta-island-do-msg-1
                 deta-island-do-msg-2))
  (premiss '(ask s-a-coastal-bars
                 deta-bar-do-msg-1
                 deta-bar-do-msg-2))
  (premiss '(ask s-a-coastal-bed-rock
                 deta-rock-do-msg-1
                 deta-rock-do-msg-2))
  )
;
(lisprule deta-shore-menu (goal shore-menu) ->
  (prog [a1]
    top (display 'deta-shore-menu-msg nil)
        (:= a1 (dread))
        (cond [(memq a1 '(0 nil no done))
               (return)]
              [(memq a1 '(1 b beach))
               (display 'deta-shore-beach-msg nil)]
              [(memq a1 '(2 br beach-ridge))
               (display 'deta-shore-beach-ridge-msg nil)]
              [(memq a1 '(3 s swale))
               (display 'deta-shore-swale-msg nil)]
              [(memq a1 '(4 sd sand sand-dune))
               (display 'deta-shore-sand-msg nil)]
              [(memq a1 '(5 p pd ponded ponded-depression))
               (display 'deta-shore-ponded-msg nil)]
              [(memq a1 '(6 i island islands))
               (display 'deta-shore-island-msg nil)]
              [(memq a1 '(7 o off off-shore-bar off-shore-bars bar bars))
               (display 'deta-shore-bar-msg nil)]
              [(memq a1 '(8 r rock rocks))
               (display 'deta-shore-rock-msg nil)]
              [t (msg t a1 " is not a legal response"
                      t "Please try again" t)]
              )
        (go top)
        )
```

```
  )
;
(lisprule deta-tidal (goal tidal-river) ->
  (display 'deta-tidal-river-msg nil)
  (cond [(is-yes (dread)) (premiss '(goal tidal-menu))])
  (display 'deta-tidal-river-do-msg nil)
  (pause)
  (cond [(gans s-a-tidal-river-islands)
         (display 'deta-tidal-island-do-msg nil)
         (pause)
         (display 'deta-tidal-island-do-msg-2 nil)])
  (display 'deta-nomore-msg nil)
  )
;
(lisprule deta-tidal-menu (goal tidal-menu) ->
  (prog [a1]
    top (display 'deta-tidal-help-msg nil)
        (:= a1 (dread))
        (cond [(memq a1 '(0 nil go go-on))
               (return)]
              [(memq a1 '(1 m mf mud mud-flats))
               (display 'deta-tidal-mud-msg nil)]
              [(memq a1 '(2 s sand sand-bars sb))
               (display 'deta-tidal-sand-msg nil)]
              [(memq a1 '(3 l levee levees))
               (display 'deta-tidal-levee-msg nil)]
              [(memq a1 '(4 i island islands))
               (display 'deta-tidal-island-msg nil)]
              [(memq a1 '(5 t tidal tidal-marshes marsh marshes))
               (display 'deta-tidal-marsh-msg nil)]
              [(memq a1 '(6 r rock rocks))
               (display 'deta-tidal-rock-msg nil)]
              [t (msg t a1 " is not a legal response"
                      t "Please try again" t)]
              )
        (go top)
        )
  )
;
;(p deta-go-rule
;   (goal deta) -->
;     (make deta start)
;     (call display deta-area-msg)
;     (call pause)
;     )
;;
;(p deta-stop-rule
;   (goal deta)
;   (deta done) -->
;     (modify 1 ^1 done)
;     (remove 2)
;     )
;;
;;      WRITE(LU(2),5)
(setq deta-area-msg
      '(t t t t " ************************************************" t
        " ****************AREA ANALYSIS****************" t
        " ************************************************" t t t t
        5 "We now begin the detailed analysis of the study area." t
```

```
        5 "Our objective is to identify and delineate all" t
        5 "landforms in the study area. The approach will be to" t
        5 "identify and delineate the easily recognizable forms" t
        5 "first. You are encouraged to review the regional" t
        5 "summary, because that information will guide our" t
        5 "detailed analysis of the study area." t t
        5 "Take a few minutes to acquaint yourself with the" t
        5 "study area using the stereoscopic aerial photography." t t
        5 "I'll Wait." t t t))
;;
;;      PAUSE 13
;;
;(p deta-split-rule-1
;   (goal deta)
;   (deta start)
;   (s-a-coastal-shoreline t) -->
;     (modify 2 ^2 shore)
;     (call display deta-shore-msg-1)
;     (call pause)
;     )
;;
;(p deta-split-rule-2
;   (goal deta)
;   (deta start)
;   (s-a-coastal-shoreline nil)
;   (s-r-coastal-shoreline-ext t) -->
;     (modify 2 ^2 shore)
;     (call display deta-shore-ext-msg)
;     )
;;
;(p deta-split-rule-3
;   (goal deta)
;   (deta start)
;   (s-a-coastal-shoreline nil)
;   (s-r-coastal-shoreline-ext nil)
;   (s-a-low-coastal-shoreline t) -->
;     (modify 2 ^2 shore)
;     (call display deta-shore-hide-msg)
;     )
;;
;(p deta-split-rule-4
;   (goal deta)
;   (deta start)
;   (s-a-coastal-shoreline nil)
;   (s-r-coastal-shoreline-ext << nil unasked >>)
;   (s-a-low-coastal-shoreline << nil unasked >>) -->
;     (modify 2 ^2 tidal-river)
;     (call display deta-shore-hide-msg)
;     )
;;
;;      IF(ISC.EQ.1)GO TO 14
;;      IF(IRCE.EQ.1)GO TO 100
;;      IF(ISLC.EQ.1)GO TO 120
;;      GO TO 200
;;
;; ------------------------------------------------------------------
;; ******************************************************************
;;  COASTLINE IN THE STUDY AREA
;; ******************************************************************
```

```
;;  ---------------------------------------------------------------
;;
;;    14 WRITE(LU(2),15)
(setq deta-shore-msg-1
      '(t t t 5 "Concentrate on the ocean shoreline. We want" t
        5 "to define the land/water boundary and then study the" t
        5 "land area adjacent to the shoreline to identify" t
        5 "landforms such as the beach, beach ridges, swales, and" t
        5 "sand dunes." t t
        5 "Now draw the boundary line along the coast separating" t
        5 "the land and water. I'll wait till you are done." t t))
;;
;;      PAUSE 14
;;
;(p deta-shore-rule-1
;    (goal deta)
;    (deta shore) -->
;      (call display deta-shore-help-msg)
;      (call yes-no shore)
;      )
;;
;(p deta-shore-talk-rule
;    (goal deta)
;    (deta shore)
;    (shore t) -->
;      (modify 2 ^2 shore-menu)
;      (remove 3)
;      )
;;
;(p deta-shore-drink-rule
;    (goal deta)
;    (deta shore)
;    (shore nil) -->
;      (modify 2 ^2 shore-do)
;      (remove 3)
;      )
;;
;;    20 WRITE(LU(2),25)
(setq deta-shore-help-msg
      '(t t 5 "Now we want to identify and delineate landforms" t
        5 "associated with the coastal shoreline. These landforms" t
        5 "will be the beach area, beach ridges, swales, sand" t
        5 "dunes, ponded depressions, islands, bars, and or rock" t t
        5 "Do you want background information on any of these" t
        5 "forms?" t t))
;;
;;      READ(LU(1),8)IANSD
;;    8 FORMAT(3A2)
;;      IF(IANSD.NE.IYES)GO TO 60
;;
;;  ---------------------------------------------------------------
;;   COASTAL LANDFORM INFORMATION MENU
;;  ---------------------------------------------------------------
;;
;(p deta-shore-menu-rule
;    (goal deta)
;    (deta shore-menu) -->
;      (call display deta-shore-menu-msg)
;      (make shore-menu (accept))
```

```
;       (call clear)
;       )
;;
;;    27 WRITE(LU(2),30)
(setq deta-shore-menu-msg
       '(t t t t t 5 "MENU FOR COASTAL LANDFORM INFORMATION:" t t
         10 "0 - No more information please" t
         10 "1 - Beach" t
         10 "2 - Beach Ridge" t
         10 "3 - Swale" t
         10 "4 - Sand Dune" t
         10 "5 - Ponded Depression" t
         10 "6 - Islands" t
         10 "7 - Off-shore Bars" t
         10 "8 - Rock" t t
         5 "Type your selection number." t t))
;;
;(p deta-shore-menu-done-rule
;    (goal deta)
;    (deta shore-menu)
;    (shore-menu << 0 nil no done >>) -->
;      (modify 2 ^2 shore-do)
;      (remove 3)
;      )
;;
;;       READ(LU(1),*)ICOAST
;;       GO TO (35,40,45,50,130,135,140,145),ICOAST
;;
;; ---------------------------------------------------------------
;;  BEACH INFORMATION
;; ---------------------------------------------------------------
;;
;(p deta-shore-menu-beach-rule
;    (goal deta)
;    (deta shore-menu)
;    (shore-menu << 1 b beach >>) -->
;      (call display deta-shore-beach-msg)
;      (modify 2 ^2 shore-menu)
;      (remove 3)
;      )
;;
;;    35 WRITE(LU(2),36)
(setq deta-shore-beach-msg
       '(t t 5 "BEACH - That area of land adjacent to the" t
         5 "shoreline containing recently reworked coastal" t
         5 "which can be wetted by storm waves." t t))
;;       GO TO 55
;;
;; ---------------------------------------------------------------
;;  BEACH RIDGE INFORMATION
;; ---------------------------------------------------------------
;;
;(p deta-shore-menu-beach-ridge-rule
;    (goal deta)
;    (deta shore-menu)
;    (shore-menu << 2 br beach-ridge >>) -->
;      (call display deta-shore-beach-ridge-msg)
;      (modify 2 ^2 shore-menu)
;      (remove 3)
```

```
;       )
;;
;;    40 WRITE(LU(2),41)
(setq deta-shore-beach-ridge-msg
      '(t t 5 "BEACH RIDGE - A ridge paralleling the present or" t
        5 "a former shoreline. Usually constructed during storm" t
        5 "periods." t t))
;;      GO TO 55
;;
;; ---------------------------------------------------------------
;;  SWALE INFORMATION
;; ---------------------------------------------------------------
;;
;(p deta-shore-menu-swale-rule
;    (goal deta)
;    (deta shore-menu)
;    (shore-menu << 3 s swale >>) -->
;      (call display deta-shore-swale-msg)
;      (modify 2 ^2 shore-menu)
;      (remove 3)
;      )
;;
;;    45 WRITE(LU(2),46)
(setq deta-shore-swale-msg
      '(t t 5 "SWALE - Local low areas behind the beach and" t
        5 "usually between beach ridges. If the bottom of the" t
        5 "swale is below the water table, the swale will contain" t
        5 "brackish water and marsh vegetation." t t))
;;      GO TO 55
;;
;; ---------------------------------------------------------------
;;  SAND DUNE INFORMATION
;; ---------------------------------------------------------------
;;
;(p deta-shore-menu-sand-rule
;    (goal deta)
;    (deta shore-menu)
;    (shore-menu << 4 sd sand sand-dune >>) -->
;      (call display deta-shore-sand-msg)
;      (modify 2 ^2 shore-menu)
;      (remove 3)
;      )
;;
;;    50 WRITE(LU(2),51)
(setq deta-shore-sand-msg
      '(t t 5 "SAND DUNE - Accumulations of wind-worked beach" t
        5 "sand in the form of low hills, irregular ridges, or" t
        5 "when large, cresent shapes." t t))
;;      GO TO 55
;;
;; ---------------------------------------------------------------
;;  PONDED DEPRESSION INFORMATION
;; ---------------------------------------------------------------
;;
;(p deta-shore-menu-ponded-rule
;    (goal deta)
;    (deta shore-menu)
;    (shore-menu << 5 p pd ponded ponded-depression >>) -->
;      (call display deta-shore-ponded-msg)
```

```
;       (modify 2 ^2 shore-menu)
;       (remove 3)
;       )
;;
;;   130 WRITE(LU(2),131)
(setq deta-shore-ponded-msg
       '(t t 5 " PONDED DEPRESSION - A large shallow ponded" t
         5 "area inland from beach. Not a swale, but usually" t
         5 "contains brackish water and marsh vegetation." t t))
;;       GO TO 55
;;
;; ---------------------------------------------------------------
;;   ISLAND INFORMATION
;; ---------------------------------------------------------------
;;
;(p deta-shore-menu-island-rule
;    (goal deta)
;    (deta shore-menu)
;    (shore-menu << 6 i island islands >>) -->
;      (call display deta-shore-island-msg)
;      (modify 2 ^2 shore-menu)
;      (remove 3)
;      )
;;
;;   135 WRITE(LU(2),136)
(setq deta-shore-island-msg
       '(t t 5 " ISLANDS - Off-shore stable land areas that can" t
         5 "survive coastal storms." t t))
;;       GO TO 55
;;
;; ---------------------------------------------------------------
;;   OFF-SHORE BAR INFORMATION
;; ---------------------------------------------------------------
;;
;(p deta-shore-menu-bar-rule
;    (goal deta)
;    (deta shore-menu)
;    (shore-menu << 7 o off off-shore-bar off-shore-bars bar bars >>) -->
;      (call display deta-shore-bar-msg)
;      (modify 2 ^2 shore-menu)
;      (remove 3)
;      )
;;
;;   140 WRITE(LU(2),141)
(setq deta-shore-bar-msg
       '(t t 5 " OFF-SHORE BARS - Off-shore deposits of" t
         5 "unconsolidated coastal materials subject to movement" t
         5 "during coastal storm periods." t t))
;;       GO TO 55
;;
;; ---------------------------------------------------------------
;;   ROCK INFORMATION
;; ---------------------------------------------------------------
;;
;(p deta-shore-menu-rock-rule
;    (goal deta)
;    (deta shore-menu)
;    (shore-menu << 8 r rock rocks >>) -->
;      (call display deta-shore-rock-msg)
```

```
;      (modify 2 ^2 shore-menu)
;      (remove 3)
;      )
;;
;;   145 WRITE(LU(2),146)
(setq deta-shore-rock-msg
      '(t t 5 "ROCK - outcropping bedrock off-shore in the the" t
        5 "area of the coastline features indicated above." t t))
;;
;(p deta-shore-menu-bad-input-rule
;    (goal deta)
;    (deta shore-menu)
;    (shore-menu <ans>) -->
;      (write (crlf) <ans> " is not a legal response"
;             (crlf) "Please try again" (crlf))
;      (call display deta-shore-beach-msg)
;      (modify 2 ^2 shore-menu)
;      (remove 3)
;      )
;;
;;    55 WRITE(LU(2),56)
(setq deta-ignored-msg
      '(t t " Type '1' if you want additional information on" t
        "  coastal landforms, otherwise type 'O'." t t))
;;       READ(LU(1),*)IDO
;;       IF(IDO.EQ.1)GO TO 27
;;
;; ------------------------------------------------------------------
;;  CONTINUING WITH IDENTIFICATION AND DELINEATION
;; ------------------------------------------------------------------
;;
;(p deta-shore-do-rule-1
;    'goal deta)
;    (deta shore-do)
;    (s-a-coastal-shoreline << nil unasked >>)
;    (s-a-coastal-shoreline-ext << nil unasked >>) -->
;      (call display deta-coast-in-sr-sa-msg)
;      (modify 2 ^2 shore-dos)
;      )
;;
;(p deta-shore-do-rule-2
;    (goal deta)
;    (deta shore-do) -->
;      (modify 2 ^2 shore-dos)
;      )
;;
;;    60 IF(ISC.NE.1)GO TO 70
;;       IF(ISCE.NE.1)GO TO 62
;;       WRITE(LU(2),61)
(setq deta-coast-in-sr-sa-msg
      '(t t 5 "The ocean coastline exists both in the study" t
        5 "region and study area and coastal features extend" t
        5 "beyond the shoreline." t t))
;;
;; ------------------------------------------------------------------
;;  BEACH IN STUDY AREA?
;; ------------------------------------------------------------------
;;
;(p deta-beach-do-rule-1
```

```
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-shoreline-ext nil)
;    (s-a-coastal-beach unasked) -->
;      (call display deta-beach-do-msg-1)
;      (call yes-no s-a-coastal-beach)
;      (remove 4)
;      )
;;
;(p deta-beach-do-rule-2
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-beach t) -->
;      (call display deta-beach-do-msg-2)
;      (call pause)
;      )
;;
;;    62 WRITE(LU(2),63)
(setq deta-beach-do-msg-1
      '(t t 5 "Does a beach exist in the study area?" t t))
;;       READ(LU(1),*)IDCB
;;       IF(IDCB.NE.1)GO TO 70
;;       WRITE(LU(2),65)
(setq deta-beach-do-msg-2
      '(t t 5 "Delineate the beach zones in the study area." t
        5 "I'll wait." t t))
;;       PAUSE 15
;;
;; ---------------------------------------------------------------
;;  BEACH RIDGES IN STUDY AREA?
;; ---------------------------------------------------------------
;;
;(p deta-beach-ridge-do-rule-1
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-beach-ridge unasked) -->
;      (call display deta-beach-ridge-do-msg-1)
;      (call yes-no s-a-coastal-beach-ridge)
;      (remove 3)
;      )
;;
;(p deta-beach-ridge-do-rule-2
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-beach-ridge t) -->
;      (call display deta-beach-ridge-do-msg-2)
;      (call pause)
;      )
;;
;;    70 WRITE(LU(2),72)
(setq deta-beach-ridge-do-msg-1
      '(t t 5 "Do beach ridges exist in the study area?" t t))
;;       READ(LU(1),*)IDCBR
;;       IF(IDCBR.NE.1)GO TO 80
;;       WRITE(LU(2),75)
(setq deta-beach-ridge-do-msg-2
      '(t t 5 "Delineate the beach ridges in the study area." t
        5 "I'll wait." t t))
;;       PAUSE 16
```

```
;;
;; ------------------------------------------------------------------
;;  SWALES IN STUDY AREA?
;; ------------------------------------------------------------------
;;
;(p deta-swale-do-rule-1
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-swale-ridge unasked) -->
;      (call display deta-swale-do-msg-1)
;      (call yes-no s-a-coastal-swale)
;      (remove 3)
;      )
;;
;(p deta-swale-do-rule-2
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-swale t) -->
;      (call display deta-swale-do-msg-2)
;      (call pause)
;      )
;;
;;    80 WRITE(LU(2),81)
(setq deta-swale-do-msg-1
      '(t t 5 "Do swales exist in the study area?" t t))
;;       READ(LU(1),*)IDCS
;;       IF(IDCS.NE.1)GO TO 90
;;       WRITE(LU(2),85)
(setq deta-swale-do-msg-2
      '(t t 5 "Delineate swales in the study area." t
        5 "I'll wait." t t))
;;       PAUSE 17
;;
;; ------------------------------------------------------------------
;;  SAND DUNES IN STUDY AREA?
;; ------------------------------------------------------------------
;;
;(p deta-sand-do-rule-1
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-sand unasked) -->
;      (call display deta-sand-do-msg-1)
;      (call yes-no s-a-coastal-sand)
;      (remove 3)
;      )
;;
;(p deta-sand-do-rule-2
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-sand t) -->
;      (call display deta-sand-do-msg-2)
;      (call pause)
;      )
;;
;;    90 WRITE(LU(2),91)
(setq deta-sand-do-msg-1
      '(t t 5 "Do sand dunes exist in the study area?" t t))
;;       READ(LU(1),*)IDCSD
;;       IF(IDCSD.NE.1)GO TO 150
```

```
;;       WRITE(LU(2),95)
(setq deta-sand-do-msg-2
      '(t t 5 "Delineate the sand dunes in the study area." t
        5 "I'll wait." t t))
;;       PAUSE 20
;;
;; ---------------------------------------------------------------
;;  PONDED DEPRESSIONS IN STUDY AREA?
;; ---------------------------------------------------------------
;;
;(p deta-ponded-do-rule-1
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-ponded-depression unasked) -->
;      (call display deta-ponded-do-msg-1)
;      (call yes-no s-a-coastal-ponded-depression)
;      (remove 3)
;      )
;;
;(p deta-ponded-do-rule-2
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-ponded-depression t) -->
;      (call display deta-ponded-do-msg-2)
;      (call pause)
;      )
;;
;;  150 WRITE(LU(2),151)
(setq deta-ponded-do-msg-1
      '(t t 5 "Are there any ponded depressions" t
        5 "in the study area?" t t))
;;       READ(LU(1),*)IDCPD
;;       IF(IDCPD.NE.1)GO TO 155
;;       WRITE(LU(2),152)
(setq deta-ponded-do-msg-2
      '(t t 5 "Delineate the ponded depressions in the study" t
        5 "area.  I'll wait." t t))
;;       PAUSE 21
;;
;; ---------------------------------------------------------------
;;  ISLANDS IN THE STUDY AREA?
;; ---------------------------------------------------------------
;;
;(p deta-island-do-rule-1
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-islands unasked) -->
;      (call display deta-island-do-msg-1)
;      (call yes-no s-a-coastal-islands)
;      (remove 3)
;      )
;;
;(p deta-island-do-rule-2
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-islands t) -->
;      (call display deta-island-do-msg-2)
;      (call pause)
;      )
```

```
;;
;;   155 WRITE(LU(2),156)
(setq deta-island-do-msg-1
      '(t t 5 "Are any of the coastal islands" t
        5 "found off-shore in the study area?" t t))
;;       READ(LU(1),*)IDCI
;;       IF(IDCI.NE.1)GO TO 160
;;       WRITE(LU(2),157)
(setq deta-island-do-msg-2
      '(t t 5 "Delineate the island-shoreline boundary of all" t
        5 "islands in the study area.  I'll wait." t t))
;;       PAUSE 22
;;
;; ---------------------------------------------------------------
;;   OFF-SHORE BARS IN COASTAL AREA?
;; ---------------------------------------------------------------
;;
;(p deta-bar-do-rule-1
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-bars unasked) -->
;      (call display deta-bar-do-msg-1)
;      (call yes-no s-a-coastal-bars)
;      (remove 3)
;      )
;;
;(p deta-bar-do-rule-2
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-bars t) -->
;      (call display deta-bar-do-msg-2)
;      (call pause)
;      )
;;
;;   160 WRITE(LU(2),161)
(setq deta-bar-do-msg-1
      '(t t 5 "Do any off-shore bars exist in" t
        5 "the study area?" t t))
;;       READ(LU(1),*)IDCOB
;;       IF(IDCOB.NE.1)GO TO 165
;;       WRITE(LU(2),162)
(setq deta-bar-do-msg-2
      '(t t 5 "Delineate the off-shore bars in the study area." t
        5 "I'll wait." t t))
;;       PAUSE 23
;;
;; ---------------------------------------------------------------
;;   BED ROCK IN THE COASTAL AREA?
;; ---------------------------------------------------------------
;;
;(p deta-rock-do-rule-1
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-bed-rock unasked) -->
;      (call display deta-rock-do-msg-1)
;      (call yes-no s-a-coastal-bed-rock)
;      (remove 3)
;      )
;;
```

```
;(p deta-rock-do-rule-2
;    (goal deta)
;    (deta shore-dos)
;    (s-a-coastal-bed-rock t) -->
;      (call display deta-rock-do-msg-2)
;      (call pause)
;      )
;;
;;   165 WRITE(LU(2),166)
(setq deta-rock-do-msg-1
      '(t t 5 "Is there bedrock in the coastal area?" t t))
;;       READ(LU(1),*)IDCRX
;;       IF(IDCRX.NE.1)GO TO 100
;;       WRITE(LU(2),167)
(setq deta-rock-do-msg-2
      '(t t 5 "Delineate the areas of outcropping bed rock in" t
        5 "the study area, while I wait." t t))
;;       PAUSE 24
;;
;;       GO TO 200
;;
;;  ------------------------------------------------------------------
;;   COASTLINE NOT IN STUDY AREA BUT COASTAL FEATURES ASSOCIATED WITH
;;   STUDY REGION COASTLINE EXTENDS INTO STUDY AREA.
;;  ------------------------------------------------------------------
;;
;;   100 WRITE(LU(2),105)
(setq deta-shore-ext-msg
      '(t t 5 "From the regional analysis we determined that" t
        5 "an ocean coastline existed in the study region but" t
        5 "not the study area, however the coastal features" t
        5 "associated with an ocean shoreline are believed to" t
        5 "extend into the study area." t t))
;;       GO TO 20
;;
;;  ------------------------------------------------------------------
;;   COASTLINE NOT IN REGION BUT COASTLINE FEATURES EXPECTED TO EXIST
;;   IN STUDY REGION AND EXTEND INTO STUDY AREA
;;  ------------------------------------------------------------------
;;
;;   120 WRITE(LU(2),125)
(setq deta-shore-hide-msg
      '(t t 5 "The coastline does not exist in the study" t
        5 "region but coastline features are expected to" t
        5 "exist in the study region and extend into the" t
        5 "study area." t t))
;;       GO TO 20
;;
;(p deta-shore-dos-to-tidal-river-rule
;    (goal deta)
;    (deta shore-dos) -->
;      (modify 2 ^2 tidal-river)
;      )
;;
;;  ------------------------------------------------------------------
;;  ******************************************************************
;;   TIDAL RIVER IN STUDY AREA
;;  ******************************************************************
;;  ------------------------------------------------------------------
```

```
;;
;(p deta-tidal-rule-1
;    (goal deta)
;    (deta tidal-river)
;    (s-a-tidal-river t) -->
;      (call display deta-tidal-river-msg)
;      (call yes-no tidal-river)
;      )
;;
;(p deta-no-tidal-rule
;    (goal deta)
;    (deta tidal-river)
;    (s-a-tidal-river nil) -->
;      (modify 2 ^2 tidal-river-island)
;      )
;;
;(p deta-tidal-rule-2
;    (goal deta)
;    (deta tidal-river)
;    (tidal-river nil) -->
;      (modify 2 ^2 tidal-river-do)
;      (remove 3)
;      )
;;
;;   200 IF(ISTR.NE.1)GO TO 1000
;;       WRITE(LU(2),205)
(setq deta-tidal-river-msg
      '(t t 5 "We now want to identify and delineate the" t
        5 "landforms associated with the tidal river in" t
        5 "the study area. This will include the river" t
        5 "shoreline, mud flats, sand bars, tidal marshes," t
        5 "natural levees, islands, and rock." t t
        5 "Do you want background information on any of these" t
        5 "forms?" t t))
;;       READ(LU(1),8)IANSD
;;       IF(IANSD.NE.IYES)GO TO 275
;;
;; ------------------------------------------------------------------
;;  TIDAL RIVER INFORMATION MENU
;; ------------------------------------------------------------------
;;
;(p deta-tidal-menu-rule
;    (goal deta)
;    (deta tidal-river)
;    (tidal-river t) -->
;      (call display deta-tidal-help-msg)
;      (make tidal-menu (accept))
;      (call clear)
;      )
;;
;;   210 WRITE(LU(2),215)
(setq deta-tidal-help-msg
      '(t t t 5 "TIDAL RIVER LANDFORM INFORMATION" t
        10 "0 - No more information" t
        10 "1 - Mud flats" t
        10 "2 - Sand bars" t
        10 "3 - Natural levees" t
        10 "4 - Islands" t
        10 "5 - Tidal marshes" t
```

```
        10 "6 - Rock" t t
        5 "Type your selection." t t))
;;      READ(LU(1),*)IRIVER
;;      GO TO(220,230,232,235,240,250),IRIVER
;;
;(p deta-tidal-menu-gone-rule
;   (goal deta)
;   (deta tidal-river)
;   (tidal-river t)
;   (tidal-menu << 0 go go-on >>) -->
;     (modify 2 ^2 tidal-river-do)
;     (remove 3 4)
;     )
;;
;; ---------------------------------------------------------------
;;  MUD FLAT INFORMATION
;; ---------------------------------------------------------------
;;
;(p deta-tidal-menu-mud-rule
;   (goal deta)
;   (deta tidal-river)
;   (tidal-river t)
;   (tidal-menu << 1 m mf mud mud-flats >>) -->
;     (call display deta-tidal-mud-msg)
;     (modify 3 ^2 t)
;     (remove 4)
;     )
;;
;;  220 WRITE(LU(2),221)
(setq deta-tidal-mud-msg
      '(t t 5 "MUD FLATS - Channel deposits of silts and clays" t
        5 "with some sands. Usually found in channel areas" t
        5 "associated with slowest currents. Will usually be" t
        5 "seen above the water surface during times of low" t
        5 "tide, but not at high tide (consult tide tables for" t
        5 "information on your particular river." t t))
;;      GO TO 260
;;
;; ---------------------------------------------------------------
;;  SAND BAR INFORMATION
;; ---------------------------------------------------------------
;;
;(p deta-tidal-menu-sand-rule
;   (goal deta)
;   (deta tidal-river)
;   (tidal-river t)
;   (tidal-menu << 2 s sand sand-bars sb >>) -->
;     (call display deta-tidal-sand-msg)
;     (modify 3 ^2 t)
;     (remove 4)
;     )
;;
;;  230 WRITE(LU(2),231)
(setq deta-tidal-sand-msg
      '(t t 5 "SAND BARS - Channel deposits of sandy and silty" t
        5 "sand materials, usually found near higher velocity" t
        5 "currents. May be exposed at low tide, but not at high" t
        5 "tide. Consult the tide tables." t t))
;;      GO TO 260
```

```
;;
;; ------------------------------------------------------------------
;;  NATURAL LEVEE INFORMATION
;; ------------------------------------------------------------------
;;
;(p deta-tidal-menu-levee-rule
;    (goal deta)
;    (deta tidal-river)
;    (tidal-river t)
;    (tidal-menu << 3 1 levee levees >>) -->
;      (call display deta-tidal-levee-msg)
;      (modify 3 ^2 t)
;      (remove 4)
;      )
;;
;;   232 WRITE(LU(2),233)
(setq deta-tidal-levee-msg
       '(t t 5 "NATURAL LEVEES - Generally fine-grained and" t
         5 "unconsolidated alluvial material deposited at the top" t
         5 "of the river bank during flood overflow periods. High" t
         5 "levee positions will only be a few inches to a few feet" t
         5 "above the surface further removed from the river." t t))
;;       GO TO 260
;;
;; ------------------------------------------------------------------
;;  ISLAND INFORMATION
;; ------------------------------------------------------------------
;;
;(p deta-tidal-menu-island-rule
;    (goal deta)
;    (deta tidal-river)
;    (tidal-river t)
;    (tidal-menu << 4 i island islands >>) -->
;      (call display deta-tidal-island-msg)
;      (modify 3 ^2 t)
;      (remove 4)
;      )
;;
;;   235 WRITE(LU(2),236)
(setq deta-tidal-island-msg
       '(t t 5 "ISLANDS - Stable land areas surrounded by" t
         5 "tidal river water." t t))
;;       GO TO 260
;;
;; ------------------------------------------------------------------
;;  TIDAL RIVER MARSH INFORMATION
;; ------------------------------------------------------------------
;;
;(p deta-tidal-menu-marsh-rule
;    (goal deta)
;    (deta tidal-river)
;    (tidal-river t)
;    (tidal-menu << 5 t tidal tidal-marshes marsh marshes >>) -->
;      (call display deta-tidal-marsh-msg)
;      (modify 3 ^2 t)
;      (remove 4)
;      )
;;
;;   240 WRITE(LU(2),241)
```

```
(setq deta-tidal-marsh-msg
      '(t t 5 "TIDAL RIVER MARSH - Shallow backwater areas" t
        5 "that flood at high tide and drain slowly as the water" t
        5 "level recedes causes swampy conditions along tidal" t
        5 "rivers as marsh grasses and other vegetation occupy" t
        5 "these sites." t t))
;;      GO TO 260
;;
;; ---------------------------------------------------------------
;;  ROCK INFORMATION
;; ---------------------------------------------------------------
;;
;(p deta-tidal-menu-rock-rule
;   (goal deta)
;   (deta tidal-river)
;   (tidal-river t)
;   (tidal-menu << 6 r rock rocks >>) -->
;     (call display deta-tidal-rock-msg)
;     (modify 3 ^2 t)
;     (remove 4)
;     )
;;
;;  250 WRITE(LU(2),251)
(setq deta-tidal-rock-msg
      '(t t 5 "ROCK - Upper reaches of rivers subjected to" t
        5 "tidal action in the Atlantic Coastal Plain should be" t
        5 "thourghly investigated for rock in the channel. This" t
        5 "suspicion results from the irregular surface of the" t
        5 "rock under the Coastal Plain sediments." t t))
;;      GO TO 260
;;
;; ---------------------------------------------------------------
;;  MORE INFORMATION?
;; ---------------------------------------------------------------
;;
;;  260 WRITE(LU(2),261)
(setq deta-ignored-msg-2
      '(t t 5 "Type '1' if you want information on other" t
        5 "tidal river forms, otherwise type 'O'." t t))
;;      READ(LU(1),*)MORE
;;      IF(MORE.EQ.1)GO TO 210
;;
;;      IF(IFLAG.EQ.1)GO TO 285
;;
;; ---------------------------------------------------------------
;;  DELINEATION OF TIDAL RIVER FEATURES
;; ---------------------------------------------------------------
;;
;(p deta-tidal-do-rule
;   (goal deta)
;   (deta tidal-river-do) -->
;     (call display deta-tidal-river-do-msg)
;     (call pause)
;     (modify 2 ^2 tidal-river-island)
;     )
;;
;;  275 WRITE(LU(2),276)
(setq deta-tidal-river-do-msg
      '(t t 5 "A tidal river exists in the study area and we need" t
```

```
        5 "to draw a boundary line along its shoreline. The object" t
        5 "will be to delineate the boundary between the tidal" t
        5 "river and all other patterns of the island." t t
        5 "Now draw the boundary line while I wait." t t))
;;
;;      PAUSE 30
;;
;;  280 WRITE(LU(2),281)
(setq deta-ignored-msg-3
      '(t t 5 "We now want to delineate the terrain units" t
        5 "associated with the tidal river. These include mud" t
        5 "flats, sand bars, natural levees, islands, tidal" t
        5 "marshes, and rock in the channel." t
        5 "Do you want background information on any of these" t
        5 "terrain units now?" t t))
;;      READ(LU(1),8)IANSD
;;      IFLAG=1
;;      IF(IANSD.EQ.IYES)GO TO 210
;;
;; ---------------------------------------------------------------
;;  DELINEATE ISLANDS IN THE STUDY AREA
;; ---------------------------------------------------------------
;;
;;  285 IF(ISTRI.NE.1)GO TO 998
;;
;(p deta-tidal-island-rule
;   (goal deta)
;   (deta tidal-river-island)
;   (s-a-tidal-river-islands t) -->
;     (call display deta-tidal-island-do-msg)
;     (call pause)
;     (call display deta-tidal-island-do-msg-2)
;     (call display deta-nomore-msg)
;     (modify 2 ^2 done)
;     )
;;
;;      WRITE(LU(2),286)
(setq deta-tidal-island-do-msg
      '(t t 5 "We now want to delineate the island or islands" t
        5 "that were found in the study area portion of the tidal" t
        5 "river. First draw a boundary between the tidal river" t
        5 "and the other patterns of an island and repeat for all" t
        5 "islands. I'll wait." t t))
;;
;;      PAUSE 31
;;
;;      WRITE(LU(2),287)
(setq deta-tidal-island-do-msg-2
      '(t t 5 "Before leaving the island/islands, scan the area" t
        5 "inside your island boundary. We will ask if other" t
        5 "terrain units, such as levees, terraces, and/or rock" t
        5 "should be delineated." t t))
;;
;(p deta-tidal-island-rule-2
;   (goal deta)
;   (deta tidal-river-island)
;   (s-a-tidal-river-islands << unasked nil >>) -->
;     (call display deta-nomore-msg)
;     (modify 2 ^2 done)
```

```
;       )
;;
;;   998 WRITE(LU(2),999)
(setq deta-nomore-msg
      '(t t 10 "THAT'S ALL THERE IS AT THIS TIME." t t t))
;;
;;
;; ---------------------------------------------------------------
;;  RETURN TO EXECUTIVE PROGRAM
;; ---------------------------------------------------------------
;;
;;  1000 CALL EXEC(8,ICALAP)
;;
;;       END
(terpr)
(princ "DETA.L Loaded")
(terpr)
```

END

10-86

DTIC